

WL-TR-93-1153

INVESTIGATION OF DRIVE-REINFORCEMENT
LEARNING AND APPLICATION OF LEARNING
TO FLIGHT CONTROL

AD-A277 442



WALTER L. BAKER (ED), STEPHEN C. ATKINS,
LEEMON C. BAIRD III, MARK A. KOENIG,
PETER J. MILLINGTON, NOEL F. NISTLER

C. S. DRAPER LABORATORY, INC
CONTROL & DECISION SYSTEMS DIV
555 TECHNOLOGY SQUARE
CAMBRIDGE, MA 02139-3563

AUGUST 1993

FINAL REPORT FOR 01/01/89-08/01/93



DTIC
ELECTE
MAR 28 1994
S F D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

94-09331



AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AFB, OH 45433-7409

DTIC ELECTE RECORDED 1

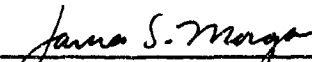
94 3 25 064


NOTICE

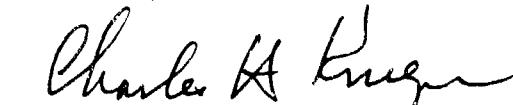
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


James S. Morgan
Project Engineer
Advanced Systems REsearch Section
Avionics Directorate
Wright Laboratory


William R. Baker, Acting Chief
Advanced Systems Research Section
Avionics Directorate
Wright Laboratory


Charles H. Krueger, Jr
Chief
Systems Avionics Division
Avionics Directorate
Wright Laboratory

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/AAAT-1, WPAFB, OH 45433-7301 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 | |
|---|---|--|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503. | | | | |
| 1. AGENCY USE ONLY (Leave blank) | | 2. REPORT DATE AUG 1993 | | 3. REPORT TYPE AND DATES COVERED FINAL 01/01/89-08/01/93 |
| 4. TITLE AND SUBTITLE INVESTIGATION OF DRIVE-REINFORCEMENT LEARNING AND APPLICATION OF LEARNING TO FLIGHT CONTROL | | | 5. FUNDING NUMBERS C F33615-88-C-1740 PE 62204 PR 2003 TA 05 WU 46 | |
| 6. AUTHOR(S) WALTER L. BAKER (ED), STEPHEN C. ATKINS, LEEMON C. BAIRD III, MARK A. KOENIG, PETER J. MILLINGTON, NOEL F. NISTLER | | | | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) C. S. DRAPER LABORATORY, INC CONTROL & DECISION SYSTEMS DIV 555 TECHNOLOGY SQUARE CAMBRIDGE, MA 02139-3563 | | | 8. PERFORMING ORGANIZATION REPORT NUMBER CSDL-R-2575 | |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AVIONICS DIRECTORATE WRIGHT LABORATORY AIR FORCE MATERIEL COMMAND WRIGHT-PATTERSON AFB, OH 45433-7409 | | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER WL-TR-93-1153 | |
| 11. SUPPLEMENTARY NOTES | | | | |
| 12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE, DISTRIBUTION IS UNLIMITED. | | | 12b. DISTRIBUTION CODE | |
| 13. ABSTRACT (Maximum 200 words) This report describes results obtained during a multiphase research program having the broad aim of investigating the application of learning systems to automatic control in general, and to flight control in particular. The first phase analyzed the drive-reinforcement learning paradigm and examined its application to automatic control, with mixed results. The second phase compared a number of alternative strategies for learning augmented control, and resulted in the conception of a new hybrid adaptive/learning control scheme. Subsequently, in the third phase, this hybrid control approach was more fully developed and applied to several nonlinear dynamical systems, including a cart-pole system, aeroelastic oscillator, and a three-degree-of-freedom aircraft. The fourth phase revisited drive-reinforcement learning from the point of view of optimal control and successfully applied a version embedded in the associative control process architecture to regulate an aeroelastic oscillator. The fifth phase examined the problem of learning augmented estimation, and resulted in the development of a preliminary estimation scheme consistent with the hybrid control approach. In the sixth and final phase, the hybrid control methodology was applied to a nonlinear, six-degree-of-freedom flight control problem, and then demonstrated via a challenging multi-axis maneuver. | | | | |
| 14. SUBJECT TERMS ACP Network Drive-Reinforcement Reinforcement Learning Adaptive Control Hybrid Control Nonlinear Control Aircraft Flight Control Learning Control Optimal Control | | | 15. NUMBER OF PAGES 502 | |
| | | | 16. PRICE CODE | |
| 17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED | 20. LIMITATION OF ABSTRACT UL | |

Table of Contents

| | | |
|-------|--|----|
| 1 | Introduction | 1 |
| 1.1 | Background: Motivation for Learning Control | 4 |
| 1.2 | Report Overview | 7 |
| 1.2.1 | Main Document | 8 |
| 1.2.2 | Attachment 1 | 9 |
| 1.2.3 | Attachment 2 | 10 |
| 1.2.4 | Attachment 3 | 11 |
| 1.3 | Program Related Technical Publications | 13 |
| 2 | Drive-Reinforcement Learning | 15 |
| 2.1 | Initial Work | 15 |
| 2.1.1 | Network Model | 15 |
| 2.1.2 | Classical Conditioning Experiments | 20 |
| 2.1.3 | Analysis | 20 |
| 2.2 | Further Work | 20 |
| 2.3 | Refinement of D-R Network Equations | 24 |
| 2.3.1 | Multiplication Learning Problem | 25 |
| 2.3.2 | Facilitatory Learning | 26 |
| 2.4 | Alternatives Strategies for Learning Control | 27 |
| 2.4.1 | Performance Measures & Design Issues | 27 |
| 2.4.2 | Alternative Approaches | 28 |
| 2.5 | D-R Revisited: ACP Networks & Learning for Optimal Control | 31 |
| 3 | Learning for Flight Control | 32 |
| 3.1 | Introduction | 32 |
| 3.2 | Background: Flight Control System Design | 34 |
| 3.2.1 | Design Difficulties | 34 |
| 3.2.2 | Traditional Design Approaches | 37 |
| 3.3 | Adaptation vs. Learning | 40 |
| 3.4 | Motivation for Hybrid Adaptive/Learning Control | 43 |
| 3.4.1 | Direct Implementation | 44 |
| 3.4.2 | Indirect Implementation | 46 |
| 3.4.3 | Summary of Hybrid Control Architectures | 48 |

| | | |
|-------|---|-----|
| 3.5 | Learning as Function Synthesis | 48 |
| 3.5.1 | Connectionist Learning Systems | 52 |
| 3.5.2 | Incremental Learning Issues | 53 |
| 3.5.3 | Spatially Localized Learning | 56 |
| 3.6 | Application Issues | 60 |
| 3.7 | Expected Benefits | 63 |
| 4 | Hybrid Adaptive/Learning Control | 65 |
| 4.1 | Controller Development | 65 |
| 4.1.1 | Baseline Linear Compensator | 66 |
| 4.1.2 | Baseline Adaptive Compensator | 69 |
| 4.1.3 | Hybrid Adaptive/Learning Compensator | 70 |
| 4.1.4 | Learning System Update | 73 |
| 4.2 | Applications of Hybrid Control to Nonlinear Systems | 74 |
| 4.2.1 | Cart-Pole System / Split-Level Track Problem | 75 |
| 4.2.2 | Aeroelastic Oscillator | 76 |
| 4.2.3 | Three-Degree-of-Freedom Flight Control | 77 |
| 4.3 | Learning Augmented Estimation | 77 |
| 5 | Multiaxis Flight Control | 82 |
| 5.1 | Control Problem Definition | 83 |
| 5.2 | Flight Simulation | 84 |
| 5.3 | Open-Loop Dynamics | 85 |
| 5.4 | Control Methodology | 86 |
| 5.4.1 | Candidate Methodologies | 87 |
| 5.4.2 | Control Methodology Selection | 92 |
| 5.5 | Controller Design | 92 |
| 5.5.1 | Control Architecture | 93 |
| 5.5.2 | Angular Rate Control | 93 |
| 5.5.3 | Sideslip Control | 100 |
| 5.5.4 | Learning System | 103 |
| 5.6 | Simulation Results | 105 |
| 5.6.1 | S-Trajectory Maneuver | 106 |
| 5.6.2 | Hybrid Control Tracking Results | 114 |
| 5.6.3 | Hybrid Control Effector Usage | 119 |
| 5.6.4 | Summary of Prediction Error Results | 124 |

| | |
|--|-----|
| 6 Conclusion | 126 |
| 6.1 Summary | 126 |
| 6.2 Recommendations for Future Work | 128 |
| 6.2.1 Reinforcement Learning Systems | 128 |
| 6.2.2 Learning Systems for Hybrid Adaptive/Learning Control | 129 |
| 6.2.3 Learning for Flight Control | 130 |
| Bibliography | 133 |
| Attachment 1 | 138 |
| Reprint of: Baird, L. (1991). <i>Learning and Adaptive Hybrid Systems for Nonlinear Control</i> , CSDL Report T-1099, M.S. Thesis, Department of Computer Science, Northeastern University. | |
| Attachment 2 | 246 |
| Reprint of: Nistler, N. (1992). <i>A Learning Enhanced Flight Control System for High Performance Aircraft</i> , CSDL Report T-1127, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T. | |
| Attachment 3 | 351 |
| Reprint of: Atkins, S. (1993). <i>Incremental Synthesis of Optimal Control Laws Using Learning Algorithms</i> , CSDL Report T-1181, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T. | |

| | |
|--------------------------------------|---|
| Accession For | |
| NTIS | CRA&I <input checked="" type="checkbox"/> |
| DTIC | TAB <input type="checkbox"/> |
| Unannounced <input type="checkbox"/> | |
| Justification | |
| By | |
| Distribution / | |
| Availability Codes | |
| Dist | Avail and/or Special |
| A-1 | |

List of Figures

| | | |
|------|--|-----|
| 1.1 | Four Control System Architectures, Employing Different Mappings | 7 |
| 2.1 | Neuronal Output Function | 19 |
| 2.2 | Reinforcement Stimuli Function | 19 |
| 2.3 | Excitatory Adaptation Function | 19 |
| 2.4 | Inhibitory Adaptation Function | 19 |
| 3.1 | Direct Adaptive/Learning Approach | 45 |
| 3.2 | Indirect Adaptive/Learning Approach | 47 |
| 3.3 | An Example of a Linear-Gaussian Network Mapping ($\mathcal{R}^2 \rightarrow \mathcal{R}$), Together with Its Underlying Influence Functions | 59 |
| 5.1 | The Basic Control Problem | 83 |
| 5.2 | Feedforward Learning Augmentation | 87 |
| 5.3 | Plant Augmentation | 89 |
| 5.4 | Model for Linear Compensator Design | 90 |
| 5.5 | Augmentation via Dynamic Inversion | 91 |
| 5.6 | Top-level Architecture of the Angular Rate CAS | 93 |
| 5.7 | The Angular Rate Control System | 94 |
| 5.8 | Proportional Plus Integral Error Dynamics | 96 |
| 5.9 | Estimation of Nonlinear Correction Terms | 99 |
| 5.10 | The β Regulator | 101 |
| 5.11 | Linear-Gaussian Network Used in the Hybrid Controller | 104 |
| 5.12 | A Snapshot of the <i>NetSim</i> Project Window Used in the 6-DOF Flight Control Demonstration | 105 |
| 5.13 | A "Side View" of the S-Trajectory Maneuver Using the Hybrid Controller | 108 |
| 5.14 | Euler Angles Associated with the S-Trajectory Maneuver Using the Hybrid Controller | 109 |
| 5.15 | S-Trajectory Using the Hybrid Controller: Altitude vs. Airspeed | 110 |
| 5.16 | S-Trajectory Maneuver Using the Hybrid Control Law: Angle-of- Attack and Sideslip vs. Time | 111 |
| 5.17 | S-Trajectory Maneuver Using the Hybrid Control Law: Load Factor vs. Time | 112 |

| | | |
|------|---|-----|
| 5.18 | S-Trajectory Maneuver Using the Hybrid Controller: Dynamic Pressure vs. Time | 113 |
| 5.19 | S-Trajectory Maneuver Under Hybrid Control: Stability-Axis Pitch Rate Tracking | 116 |
| 5.20 | S-Trajectory Maneuver Under Hybrid Control: Stability-Axis Roll Rate Tracking | 117 |
| 5.21 | S-Trajectory Maneuver Under Hybrid Control: Stability-Axis Yaw Rate Tracking | 118 |
| 5.22 | S-Trajectory Maneuver Under Hybrid Control: Aileron Response | 120 |
| 5.23 | S-Trajectory Maneuver Under Hybrid Control: Symmetric Stabilator Response | 121 |
| 5.24 | S-Trajectory Maneuver Under Hybrid Control: Differential Stabilator Response | 122 |
| 5.25 | S-Trajectory Maneuver Under Hybrid Control: Rudder Response | 123 |
| 5.26 | Summary of Prediction Errors Using Adaptive, Hybrid Adaptive/Learning, and "Ideal" Augmentation | 125 |

List of Tables

| | | |
|-----|---|-----|
| 4.1 | Summary of Three Related Model-Reference Compensators | 73 |
| 5.1 | Modal Frequencies for $H = 5,000$ ft and $V = 600$ ft/s | 86 |
| 5.2 | Main <i>NetSim</i> Component Modules for S-Trajectory Demonstration | 106 |
| 5.3 | S-Trajectory (Open-Loop) Guidance Commands | 106 |

Acknowledgment

This document is the final technical report for the research program entitled *Investigation of Drive-Reinforcement Learning & Application of Learning to Flight Control*, conducted by the Charles Stark Draper Laboratory, Inc., Cambridge, MA, under contract to the United States Air Force Wright Laboratory, Wright-Patterson AFB, OH (Contract No. F33615-88-C-1740).

The authors are indebted to a multitude of individuals for their support and many contributions during the course of this research program. Bill Goldenthal played a crucial role in developing and writing the original proposal. Paul Motyka and Milt Adams also contributed to the proposal process and were both instrumental in getting the program off to a good start; in addition, they (along with Bill Bonnice) helped maintain the program on a steady course throughout its tenure. Jeff Alexander developed a wonderful and sophisticated simulation package (from scratch) that facilitated our work greatly and proved to have many other uses and features (planned or otherwise!). As always, Jim Harrison provided sound advice, a patient ear, and continual encouragement—he helped us strive for more. Jay Farrell contributed generally to the understanding and development of learning systems for control. Bill Bonnice contributed to technical discussions and, perhaps more importantly, had strong faith in our abilities and in the promise of our work.

From the outset, Harry Klopff offered us two key things: (i) a new, biologically motivated, perspective from which to consider adaptive and learning systems, and (ii) early enunciation and advocacy for the central role of feedback in such systems. Jim Morgan and Mark Mears both deserve special credit for their great patience and staunch support in the face of many contractual, technical, and programmatic twists and turns. Mark contributed to several valuable technical discussions, and was particularly helpful in identifying different roles for learning in the context of flight control. Harry and Jim contributed to numerous technical discussions and helped us to understand reinforcement learning. Jim did an excellent job of monitoring our progress, helping to resolve difficulties as they arose, and generally keeping the program moving forward. Moreover, Jim's great enthusiasm for our work touched us all and made all the extra effort worthwhile.

1 Introduction

A synopsis of this research program is provided below. General motivation for the use of learning in control is provided as background material in Section 1.1. An overview of this report follows in Section 1.2. A list of technical publications produced during the course of this work is provided in Section 1.3.

This report describes results obtained during a multiphase research program having the broad aim of investigating the application of learning systems to automatic control in general, and to flight control in particular. The first phase analyzed the original drive-reinforcement learning paradigm [Klopf (1988)] and examined its application to automatic control, with mixed results. The second phase compared a number of alternative control strategies including conventional linear control [Friedland (1986)], adaptive control [Åström & Wittenmark (1989)], and other reinforcement learning control methods [Barto, Sutton, & Anderson (1983)], and resulted in the conception of a new hybrid adaptive/learning control scheme [Baker & Farrell (1990)]. Subsequently, in the third phase, this hybrid control approach was more fully developed and applied to several nonlinear dynamical systems, including a cart-pole system, aeroelastic oscillator, and a three-degree-of-freedom high performance aircraft. The fourth phase revisited drive-reinforcement learning from the point of view of optimal control and successfully applied a version embedded in the associative control process architecture [Klopf, Morgan, Weaver (1992)] to regulate an aeroelastic oscillator. The fifth phase examined the problem of learning augmented estimation, and resulted in the development of a preliminary estimation scheme consistent with the hybrid adaptive/learning control approach. In the sixth and final phase, the hybrid control methodology was applied to a nonlinear, six-degree-of-freedom flight control problem, and then successfully demonstrated via a challenging multiaxis maneuver.

Initial work with the basic drive-reinforcement (D-R) learning algorithm showed considerable promise for its application to automatic control. However, it was soon demonstrated that without added functionality the basic algorithm could not serve alone as a learning controller, at least not in the usual sense of what is meant by learning control. Moreover, an examination of a number of alternative

strategies (including other reinforcement learning strategies) revealed many candidates with both advantages and disadvantages, but none with a clear dominance over the others (particularly in the context of flight control). During this period, a novel hybrid adaptive/learning control scheme [Baird & Baker (1990); Baker & Farrell (1990)] was conceived that provided many of the advantages seen among the candidates considered, yet that avoided many of their disadvantages. In light of this, a decision was made to pursue this new approach in lieu of others. At the same time, emphasis was placed on the use of learning to address problems in control related to uncertainty and nonlinearity, rather than to problems related to optimization and implicit behavioral objectives.

Accordingly, development and refinement of the hybrid adaptive/learning control methodology continued during a substantial portion of the program. This approach was successfully applied to a number of nonlinear dynamical systems, culminating in its application to a multiaxis flight control problem. The learning augmented flight control system was constructed by augmenting a simple linear compensator design with both an adaptive and a learning capability. The model-based *linear* compensator was designed following a procedure similar to that described in [Anderson & Schmidt (1991)]. The *adaptive* compensator was developed by incorporating and extending ideas presented in [Yucef-Toumi & Ito (1990)]. Finally, a *hybrid* adaptive/learning control system was developed by combining the same adaptive compensator with a spatially localized learning system based on a linear-Gaussian network [Baker & Farrell (1990); Millington (1991)]. The hybrid adaptive/learning flight control system was successfully demonstrated on the 6-DOF nonlinear aircraft model via a challenging multiaxis maneuver. To illustrate the benefit of learning augmentation, the basic linear and adaptively augmented compensator designs were used as baseline controllers.

In the latter part of the program, a decision was made to revisit, from the point of view of optimal control, the D-R learning paradigm and, more generally, the new associative control process (ACP) architecture [Klopf, Morgan, Weaver (1992)] in which it was embedded. It was found that the ACP architecture provided the additional functionality needed by the original D-R algorithm to allow it to be used for optimal control. Although it was too late in the program timeline to consider its application to flight control, an ACP-based controller was developed and suc-

cessfully applied to the problem of regulating the output of a nonlinear aeroelastic oscillator model, in an optimal fashion.

Throughout this research program, software development was substantially facilitated through the use of a custom simulation environment known as *NetSim* [Alexander, et al. (1991)] that was designed especially for the investigation of connectionist network based learning systems, and also by the existence of a representative high performance nonlinear aircraft model in FORTRAN [Brumbaugh (1990)]. New software development was essentially limited to the creation of *NetSim* modules for various example applications and to the conversion of the FORTRAN aircraft model into a C-based *NetSim* module. In addition, further development and refinement of these modules and of the *NetSim* application was performed.

The aircraft code used in this work was derived from a six-degree-of-freedom (6-DOF) high performance aircraft model, incorporating nonlinear aerodynamic effects (based on empirically derived tabular data), nonlinear engine dynamics, and nonlinear actuator dynamics (including rate and position limits). This code is a slightly modified version of an F-15 simulation developed by NASA/Dryden. A more detailed description of the basic aircraft model and its FORTRAN implementation can be found in [Brumbaugh (1990)].

Results obtained during this research program clearly demonstrate many of the potential benefits of learning augmented control and especially the advantages that may be gained in terms design facilitation, automatic accommodation of uncertainty, on-line performance optimization, and operational efficiency. The bottom line is that *learning augmentation is beneficial to automatic control in general and to flight control, in particular.*¹ Although significant progress was made during this research program, these results also serve to indicate that further work is needed. Topics for future research and development include:

- further development of the hybrid control and estimation methodology

¹ This claim is further supported by a second research program funded by the Navy (under USN Contract No. N62269-91-C-0033) which involved the application of the hybrid adaptive/learning control technique *conceived and developed in this program* to a full subsonic envelope, handling qualities improvement system for a high performance aircraft [Millington & Baker (1992)].

- development and refinement of variable structure learning systems
- further investigation of reinforcement learning in the context of optimal control and multiplayer game problems
- research and development of continuous input/output reinforcement learning systems

1.1 Background: Motivation for Learning Control

Advanced control systems for autonomous or highly automated systems are expected to maintain closed-loop system stability and performance over a wide range of operating conditions and events. This objective can be difficult to achieve due to the complexity of both the plant (i.e., the system to be controlled) and the performance objectives, and due to the presence of uncertainty. Such complications may result from nonlinear or time-varying behavior, poorly modeled plant dynamics, high dimensionality, multiple inputs and outputs, complex objective functions, operational constraints, imperfect measurements, and the possibility of actuator, sensor, or other component failures. Each of these effects, if present, must be addressed if the system is to operate reliably in an automatic fashion. A view strongly advocated here is that *learning control systems* may be used advantageously to address several of these difficulties. In particular, in this research program we have focused on the control of complex dynamical systems that are *poorly modeled and nonlinear*.

Unfortunately, it is difficult to provide a precise and completely satisfactory definition for the term "learning control system." One interpretation that is, however, consistent with the prevailing literature (e.g., [Klopf & Morgan (1990)]) is that:

*A learning control system is one that has the ability to improve its performance in the future, based on experiential information it has gained in the past, through closed-loop interactions with the plant and its environment.*¹

¹ To help focus the discussion that follows and avoid any unnecessary controversy, we will further limit our subject to include, primarily, the type of learning that one might associate with sensorimotor control, and exclude more sophisticated learning behaviors (e.g., planning and exploration).

There are several implications of this statement. One implication is that a learning control system has some *autonomous* capability, since it has the ability to improve its own performance. Another is that it is *dynamic*, since it may vary over time. Yet another implication is that it has *memory*, since it can exploit past experience to improve future performance. Finally, to improve its performance, the learning system must operate in the context of an *objective function* and, moreover, it must receive *performance feedback* that characterizes the appropriateness of its current behavior in that context.

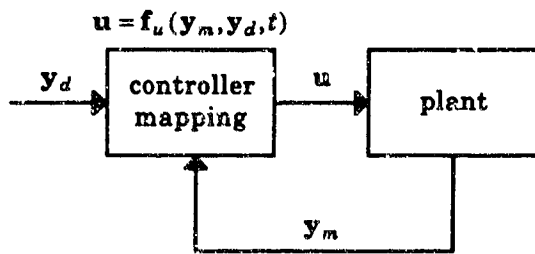
In a fundamental sense, the control design problem is to find an appropriate mapping, from measured plant outputs y_m and desired plant outputs y_d , to a control action u that will produce satisfactory behavior in the closed-loop system. In other words, the problem is to choose a function (a *control law*) $u = k(y_m, y_d, t)$ that achieves certain performance objectives when applied to the open-loop system. In turn, the solution to this problem may naturally involve other mappings; e.g., a mapping from the current plant operating condition to the parameters of a controller or local plant model, or a mapping from measured plant outputs to estimated plant state. Accordingly, *a learning system that could be used to synthesize such mappings on-line would be an advantageous component of an advanced control system.* To successfully employ learning systems in this manner, one must have an effective means for their implementation and incorporation into the overall control system architecture. The belief that connectionist systems offer a suitable means with which to implement learning control systems has been the impetus for a large body of recent research¹ (e.g., [Albus (1975); Anderson (1989); Atkins (1993); Baird & Baker (1990); Baird (1991); Baker & Farrell (1990, 1991, 1992); Baker & Millington (1992, 1993); Barto, Sutton, & Anderson (1983); Berger (1992); Cerrato (1993); Farrell & Baker (1991, 1992, forthcoming); Klopff & Morgan (1990); Klopff, Morgan, & Weaver (1992); Millington & Baker (1992); Millington (1991); Millington, Baker, & Koenig (1993); Morgan, Patterson, & Klopff (1990); Nistler (1992); Steinberg (1992); Vos, Baker, & Millington (1991)]). Perhaps a more cogent statement of affairs is that, in the context of control, learning can be viewed as the automatic incremental synthesis of multivariable functional mappings and, moreover, that connectionist systems provide a useful framework for realizing such mappings.

¹ This is not intended to be a comprehensive list.

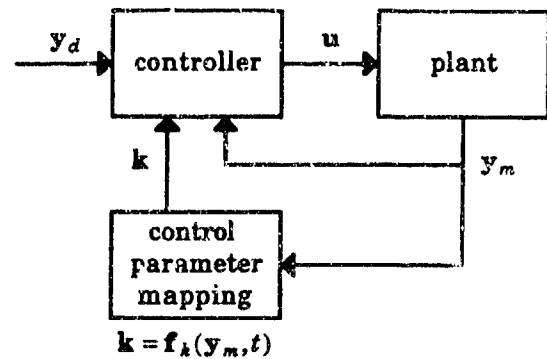
The necessity for applying learning arises in situations where a system must operate in conditions of uncertainty, and when the available *a priori* information is so limited that it is impossible or impractical to design in advance a system that has fixed properties and also performs sufficiently well [Tsypkin (1973)]. In the context of intelligent control, learning can be viewed as a means of solving those problems that lack sufficient *a priori* information to allow a complete and *fixed* (i.e., nonadaptable) control system design to be derived in advance. Thus, a central role of learning in intelligent control is to enable a wider class of problems to be solved, by reducing the prior uncertainty to the point where satisfactory solutions can be obtained *on-line*. This result is achieved empirically, by means of performance feedback, association, and memory (or knowledge base) adjustment.

One of the principal benefits of learning control, given the present state of its technological development, derives from the ability of learning systems to automatically synthesize mappings that can be used advantageously within a control system architecture. Examples of such mappings include a *controller* mapping that relates measured and desired plant outputs to an appropriate set of control actions (Fig. 1.1a), a related *control parameter* mapping that generates parameters (e.g., gains) for a separate controller (Fig. 1.1b), a *model state* (or *estimator*) mapping that produces state estimates (Fig. 1.1c), and a *model parameter* mapping that relates the plant operating condition to an accurate set of model parameters (Fig. 1.1d). In general, these mappings may represent dynamic functions (i.e., functions that involve temporal differentiation or integration).

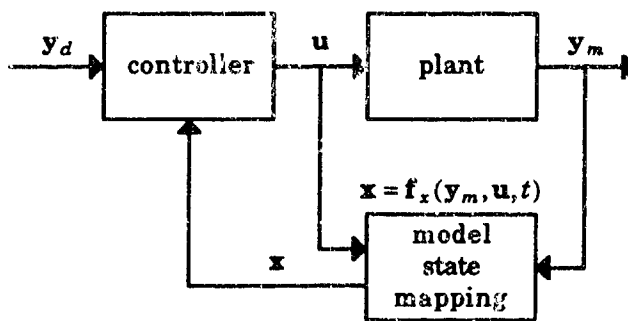
Learning is required when these mappings cannot be determined completely in advance because of *a priori* uncertainty (e.g., modeling error). In a typical learning control application, the desired mapping is *stationary* (i.e., does not depend explicitly on time), and is expressed (implicitly) in terms of an objective function involving the outputs of both the plant and the learning system. The objective function is used to provide *performance feedback* to the learning system, which must then *associate* this feedback with specific *adjustable* elements of the mapping that is currently stored in its *memory*. The underlying idea is that *experience* can be used to improve the mapping furnished by the learning system.



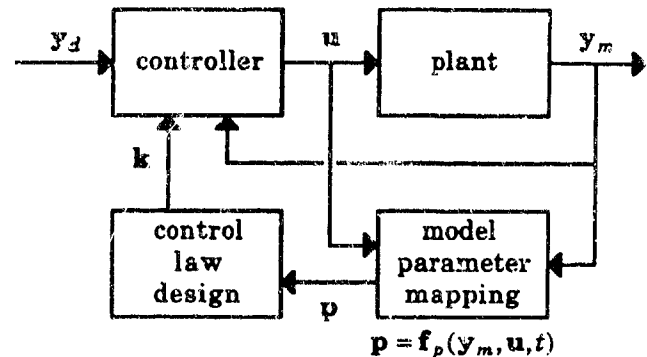
(a) controller mapping



(b) control parameter mapping



(c) model state (estimator) mapping



(d) model parameter mapping

Figure 1.1. Four Control System Architectures, Employing Different Mappings.

1.2 Report Overview

The remainder of this report is organized into a number of chapters (described below) whose subjects reflect the principal tasks pursued under this research program. In addition, three graduate student theses (also described below) are included in their entirety as attachments. A total of 14 technical publications were generated based on work that was performed during the course of this program (see Section 1.3). To minimize redundancy as well as the cost of producing this final report, extensive reference will be made to the relevant parts of these documents and to certain articles contained in the bibliography.

1.2.1 Main Document

The main document presents an overview of this research program, including a summary of all activities and accomplishments. The principal themes of the remaining chapters are outlined below.

Chapter 2: Drive-Reinforcement Learning addresses those aspects of the program that were primarily concerned with an investigation of reinforcement learning methods and their application to problems in automatic control. The focus of this part of the investigation was on *drive-reinforcement learning* [Klopf (1988)] and on *associative control process* networks [Klopf, Morgan, & Weaver (1992); Baird & Klopf (1992)]. Conventional alternatives to learning control are also reviewed. In addition, a reinforcement learning system based on the use of the *associative search element / adaptive critic element* [Barto, Sutton, & Anderson (1983)] was examined. Experimental results were obtained by applying some of these approaches to a cart-pole stabilization and tracking problem. The use of reinforcement learning in the context of optimal control is also examined.

Chapter 3: Learning for Flight Control provides a high-level discussion of the motivation for, as well as the issues underlying, the use of learning in flight control applications. Based on this investigation, new hybrid adaptive/learning control architectures are conceived.

Chapter 4: Hybrid Adaptive/Learning Control provides a detailed mathematical development of a novel hybrid adaptive/learning control methodology. In addition, a preliminary technique for learning augmented estimation which is consistent with the hybrid control architecture is also presented.

Chapter 5: Multiaxis Flight Control addresses those aspects of the program related to the development and demonstration of a learning augmented flight control system for a nonlinear vehicle model representative of a modern high performance aircraft. A challenging, multiaxis "S trajectory" maneuver is used to highlight the benefits of learning augmentation.

Chapter 6: Conclusion provides a summary of this program and recommendations for future research.

A bibliography of the references used in the course of this research is included at the end of this document. Additional bibliographies are included at the end of each attachment.

1.2.2 Attachment 1

Attachment 1 is a Master's Thesis, entitled *Learning and Adaptive Hybrid Systems for Nonlinear Control* [Baird (1991)], that was completed under this research program. The object of this thesis was to find methods for combining learning systems with adaptive systems so as to achieve good control in the presence of both spatial and temporal functional dependencies. Several methods were developed for augmenting the estimation carried out by an indirect adaptive system with the additional information available from a learning system. In addition to developing a simple form of learning augmented estimation, various issues in the construction and use of connectionist learning systems were explored in this context.

Chapter 2: Background outlines some of the important concepts and historical development of connectionist learning systems, control systems, and approaches for using connectionist learning systems for control.

Chapter 3: Hybrid Control Architecture covers the adaptive controller and connectionist networks that were integrated into a single hybrid controller. Both the individual components and the final, integrated system, are described in detail.

Chapter 4: Connectionist Learning for Control covers some of the difficulties associated with learning systems used for control, and describes various methods that might be used to address those difficulties.

Chapter 5: Experiments describes the various simulations performed. These results are presented graphically and are interpreted in relation to the original research goals.

Chapter 6: Conclusions and Recommendations summarizes what has been accomplished, draws conclusions, and points out areas in which future research should be focused.

1.2.3 Attachment 2

Attachment 2 is a Master's Thesis, entitled *A Learning Enhanced Flight Control System for High Performance Aircraft* [Nistler (1992)], that was completed under this research program. This thesis explored the use of a learning system to augment an adaptive flight controller. The extent to which learning can be used to improve an adaptive flight control system architecture, as well as the difficulties introduced by learning augmentation, were examined. The primary objective of this thesis was to illustrate the advantages of a hybrid adaptive/learning control system in terms of its ability to accommodate unmodeled dynamics and to reduce state-dependent uncertainties in the system model. This hybrid approach offers advantages over conventional techniques in terms of performance, robustness, and design refinement costs.

Chapter 2: Background discusses some of the challenges associated with flight control law design. Moreover, background information on traditional control techniques is provided to serve as a foundation for the hybrid control law development, and also as a basis for comparison of alternative designs. The theoretical concepts underlying connectionist learning systems, as well as some approaches to using learning systems for control, are also presented.

Chapter 3: Technical Approach considers technical aspects of the hybrid control law. This is accomplished by first presenting the underlying theory of the adaptive system and the spatially localized learning system before moving on to a derivation of the hybrid system. General characteristics of the hybrid controller are also presented.

Chapter 4: Experiments presents two examples to illustrate the implementation and performance of the hybrid control law. The first experiment used the hybrid system to control a relatively simple nonlinear aeroelastic oscillator. Due to the low dimensionality of the plant, and the availability of a truth model, the analysis and evaluation of the hybrid control system for the aeroelastic oscillator was greatly simplified. In the second experiment, the hybrid system was applied to a realistic high performance aircraft model. Descriptions of the major components of the aircraft model as well as its significant control characteristics are also provided. An evaluation of aircraft performance when controlled by the hybrid sys-

tem is presented and compared with other designs for various simulations. Learning system characteristics are also described.

Chapter 5: Conclusions and Recommendations summarizes the major contributions of this thesis. In addition, recommendations for future research are presented.

1.2.4 Attachment 3

Attachment 3 is a Master's Thesis, entitled *Incremental Synthesis of Optimal Control Laws Using Learning Algorithms* [Atkins (1993)], that was completed under this research program. The primary objective of this thesis was to incrementally synthesize a nonlinear optimal control law, through real-time, closed-loop interactions between the dynamic system, its environment, and a learning system, when substantial initial model uncertainty exists. The dynamic system is assumed to be nonlinear, time-invariant, and of known state dimension, but otherwise only inaccurately described by an *a priori* model. The problem, therefore, requires either explicit or implicit system identification. No disturbances, noise, or other time-varying dynamics were assumed to exist. The optimal control law is assumed to extremize an evaluation of the state trajectory and the control sequence, for any initial condition.

One goal of this thesis was to present an investigation of several approaches for incrementally synthesizing (on-line) an optimal control law. A second goal was to propose a *direct/indirect* framework, with which to distinguish such architectures. This thesis unifies a variety of concepts from control theory and behavioral science (where the learning process has been considered extensively) by presenting two different learning algorithms applied to the same control problem: the Associative Control Process (ACP) algorithm [Klopf, Morgan, & Weaver (1992)], which was initially developed to predict animal learning behavior, and *Q* learning [Watkins (1989)], which derives from the mathematical theory of value iteration.

Chapter 2: The Aeroelastic Oscillator describes a two-state physical system that exhibits interesting nonlinear dynamics, and was used throughout the thesis to evaluate different control algorithms that incorporate learning. The algorithms that are explored in Chapters 3-5 do not explicitly employ dynamic models of the

system and, therefore, may be categorized as *direct* methods of learning an optimal control law. In contrast, Chapter 6 develops an *indirect*, model-based, approach to learning an optimal control law.

Chapter 3: The Associative Control Process reviews the ACP learning paradigm (including drive-reinforcement learning) and discusses an application of an ACP network to an optimal control problem involving the regulation of a nonlinear aeroelastic oscillator. Simulation results are presented. This chapter also introduces the concept of direct learning methods in conjunction with the on-line synthesis of an optimal control law.

Chapter 4: Policy and Value Iteration reviews a number of basic concepts including those of policy iteration, value iteration, and Q learning. In addition, simulation results of the application of Q learning to the aeroelastic oscillator problem are presented.

Chapter 5: Temporal Difference Methods reviews a general theory of temporal difference methods as developed in [Sutton (1988)]. Following this review, a comparison of the preceding direct methods for the synthesis of optimal control laws is presented.

Chapter 6: Indirect Learning Optimal Control introduces the notion of indirect methods in the on-line synthesis of optimal control laws and derives several that are optimal with respect to various finite horizon cost functionals. The structure of the control laws with and without learning augmentation is presented for several cost functionals, to illustrate the manner in which learning may be used.

Chapter 7: Summary reviews the major contributions of this thesis; in addition, recommendations for future research are presented.

Appendix A: Differential Dynamic Programming briefly reviews both dynamic programming (DP) and differential dynamic programming (DDP), which are classical, alternative methods for synthesizing optimal controls. DDP is not restricted to operations over a discrete input space and discrete output space. The DP and DDP algorithms are model-based and, therefore, learning may be introduced by explicitly improving the *a priori* model, resulting in an indirect learning

optimal controller. However, neither DP nor DDP is easily implemented on-line. Additionally, DDP does not address the problem of synthesizing a control law over the full state-space.

1.3 Program Related Technical Publications

The work underlying the following list of technical publications was performed, in whole or in part, under this research program. The publications are grouped according to type (i.e., graduate student thesis, conference paper, or book chapter) and are listed in chronological order within each group. The research performed in conjunction with the three graduate student theses was fully funded by this program, while only partial support was provided in the case of each of the remaining publications. Note that each thesis is included in its entirety as an attachment to this document.

Graduate Student Theses

- Baird, L. (1991). *Learning and Adaptive Hybrid Systems for Nonlinear Control*, CSDL Report T-1099, M.S. Thesis, Department of Computer Science, Northeastern University. [Attachment 1]
- Nistler, N. (1992). *A Learning Enhanced Flight Control System for High Performance Aircraft*, CSDL Report T-1127, M.S. Thesis, Department of Aeronautics and Astronautics, M.I.T. [Attachment 2]
- Atkins, S. (1993). *Incremental Synthesis of Optimal Control Laws Using Learning Algorithms*, CSDL Report T-1181, M.S. Thesis, Department of Aeronautics and Astronautics, M.I.T. [Attachment 3]

Conference Papers

- Baird, L. & Baker, W. (1990). "A Connectionist Learning System for Nonlinear Control," proceedings, *1990 AIAA Conference on Guidance, Navigation, and Control*.
- Baker, W. & Farrell, J. (1990). "Connectionist Learning Systems for Control," proceedings, *SPIE OE/Boston '90*.
- Farrell, J. & Baker, W. (1991). "Learning Augmented Control for Advanced Autonomous Underwater Vehicles," proceedings, *18th Annual AUVS Technical Symposium and Exhibit*.
- Alexander, J., Baird, L., Baker, W., & J. Farrell (1991). "A Design & Simulation Tool for Connectionist Learning Control Systems: Application to Autonomous

Underwater Vehicles," proceedings, *1991 SCS Summer Computer Simulation Conference*.

Baker, W. & Farrell, J. (1991). "Learning Augmented Flight Control for High Performance Aircraft," proceedings, *1991 AIAA Conference on Guidance, Navigation, and Control*.

Vos, D., Baker, W., & Millington, P. (1991). "Learning Augmented Gain Scheduling Control," proceedings, *1991 AIAA Conference on Guidance, Navigation, and Control*.

Baker, W. & Millington, P. (1992). "Adaptation & Learning in Control Systems, Application to Flight Control," proceedings, *1992 Government Neural Network Applications Workshop*.

Millington, P., Baker, W., & Koenig, M. (1993). "Control Augmentation System (CAS) Synthesis via Adaptation & Learning," proceedings, *1993 AIAA Conference on Guidance, Navigation, and Control*.

Book Chapters

Baker, W. & Farrell, J. (1992). "An Introduction to Connectionist Learning Control Systems," in White, D. & Soige, D., eds., *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold.

Farrell, J. & Baker, W. (1993). "Learning Control Systems," in Antsaklis, P. & Passino, K., eds., *Intelligent and Autonomous Control Systems*, Kluwer Academic.

Farrell, J. & Baker, W. "Learning Control Systems: Motivation and Implementation," to appear in *Intelligent Control Systems: Theory and Practice*, Gupta, M. & Sinha, N., eds., IEEE Press.

2 Drive-Reinforcement Learning

This chapter addresses those aspects of the program that were primarily concerned with an investigation of reinforcement learning methods and their application to problems in automatic control. The initial focus of this part of the investigation was on the *drive-reinforcement* (D-R) *learning* paradigm [Klopf (1988)]; later on, the scope was expanded to include *associative control process* (ACP) networks [Klopf, Morgan, & Weaver (1992); Baird & Klopf (1992)]. Conventional alternatives to learning control were also reviewed. In addition, a reinforcement learning based on the use of the *associative search element / adaptive critic element* [Barto, Sutton, & Anderson (1983)] was examined. Experimental results were obtained by applying some of these approaches to a cart-pole stabilization and tracking problem. The use of reinforcement learning and related methods (e.g., ACP networks [Klopf, Morgan, & Weaver (1992)], temporal difference methods [Sutton (1988)], and *Q* learning [Watkins (1989)]) in the context of optimal control was also examined.

2.1 Initial Work

The first part of our investigation of the drive-reinforcement learning paradigm amounted to a review of the relevant technical literature on the subject, preliminary theoretical analysis of the algorithm, and an experimental study of the behavior of the algorithm via a computer simulation we developed. As a check of the validity of this software simulation, we successfully duplicated every experimental result described in [Klopf (1988)]. The motivation and development of the D-R learning paradigm as presented in [Klopf (1988)] is quite lucid and has no worthy substitute—the interested reader is strongly encouraged to examine this reference, as well as [Klopf, Morgan, & Weaver (1992)], before considering the rest of this chapter. Thus, we will not provide a summary of these works per se, although some background information may be found in Attachment 3.

2.1.1 Network Model

Early on in our investigation, it seemed likely that a *network* of drive-reinforcement learning neurons would be a useful and perhaps even necessary

development for general applications of this paradigm to problems in automatic control. To this end, we set out to develop a suitably general dynamic network model, with the following objectives in mind:

- develop a general model of a D-R learning network
- emphasize adaptive/learning control application
- maintain fidelity to Klopfs (single) neuronal model

The drive-reinforcement learning algorithm defined in [Klopf (1988)] really only pertains to the behavior of a single neuron—it is not immediately clear how one should generalize such a model to describe the behavior of a network of interacting drive-reinforcement neurons.

A number of interesting design issues arise when one contemplates the extension of a single D-R neuron to a network of many such units. Some of these network design issues are listed below:

- number of neurons (size of network)
- primary drive selection
- "potential" (acquired) drive selection
- network inputs
 - binary
 - real-valued
- parameters
 - learning rate coefficients
 - learning interval

The proposed network model outlined below appears satisfactory for two important reasons: (i) in the special case where the network is comprised of exactly one neuron, the network model corresponds exactly with [Klopf (1988)] and (ii) the structure of the network model strongly resembles the structure of an adaptive controller implemented as a system of nonlinear difference equations.

Network Drive Equations

The network drive equations provide a mathematical description of the input-output behavior of a drive-reinforcement learning network. These two equations are modeled after the usual state-space representation of a dynamic system. The first

equation introduces the concept of state [the vector $\mathbf{x}(k)$] to drive-reinforcement learning. This equation describes the causal relationship between the current state of the network $\mathbf{x}(k)$ and its previous state $\mathbf{x}(k-1)$ and current input stimuli $\mathbf{u}_{cs}(k)$ and $\mathbf{u}_{us}(k)$. The second equation simply describes the (linear) mapping between the current state of the network $\mathbf{x}(k)$ and its outputs $\mathbf{y}(k)$.

$$\begin{aligned}\mathbf{x}(k) &= \mathbf{f}_N[\{\mathbf{A}^+(k) + \mathbf{A}^-(k) + \mathbf{A}^0\} \cdot \mathbf{x}(k-1) + \{\mathbf{B}^+(k) + \mathbf{B}^-(k)\} \cdot \mathbf{u}_{cs}(k) + \mathbf{B}^0 \cdot \mathbf{u}_{us}(k)] \\ \mathbf{y}(k) &= \mathbf{C}^0 \cdot \mathbf{x}(k)\end{aligned}$$

Note the role that each matrix plays in determining the input-output behavior of the drive-reinforcement learning network. The matrices $\mathbf{A}^+(k)$, $\mathbf{A}^-(k)$, and \mathbf{A}^0 characterize those interactions that are wholly internal; that is, those interactions that occur among the neurons of the network. The matrices $\mathbf{B}^+(k)$, $\mathbf{B}^-(k)$, and \mathbf{B}^0 map the inputs to the network state, while the matrix \mathbf{C}^0 maps the network state to the outputs.

The elements of the matrices $\mathbf{A}^+(k)$, $\mathbf{A}^-(k)$, $\mathbf{B}^+(k)$, and $\mathbf{B}^-(k)$ represent plastic synaptic efficacies that are constrained to be strictly and exclusively excitatory, inhibitory, or nonactive; i.e.,

$$\begin{aligned}\{a_{ij}^+(k) \geq w_{min} \text{ and } a_{ij}^-(k) \leq -w_{min}\} \text{ or } \{a_{ij}^+(k) = a_{ij}^-(k) = 0\} \text{ for all } k \\ \{b_{ij}^+(k) \geq w_{min} \text{ and } b_{ij}^-(k) \leq -w_{min}\} \text{ or } \{b_{ij}^+(k) = b_{ij}^-(k) = 0\} \text{ for all } k\end{aligned}$$

where w_{min} is the minimum allowable absolute value for all active plastic excitatory and inhibitory synaptic efficacies. The matrices \mathbf{A}^0 , \mathbf{B}^0 , and \mathbf{C}^0 are all constant; the elements of these matrices may be negative, positive, or zero.

The vector-valued function $\mathbf{f}_N(\mathbf{x})$ is defined below:

$$\mathbf{f}_N(\mathbf{x}) = [f_N(x_1), f_N(x_2), \dots, f_N(x_n)]^T$$

where $f_N(\cdot)$ is the neuronal output function shown in Fig. 2.1 and the symbol " T " denotes the matrix (or vector) transpose operation.

Network Reinforcement Equations

The network reinforcement equations describe the adaptive behavior of a drive-reinforcement learning network. The first pair of equations account for adaptation due to external reinforcement, while the second pair account for internal (inter-neuron or intranetwork) reinforcement.

$$\begin{aligned}
b_{ij}^+(k+1) &= f_w \left[b_{ij}^+(k) + \{x_i(k) - x_i(k-1)\} \sum_{l=1}^{\tau} \alpha_l b_{ij}^+(k-l) f_s[u_{cs,j}(k-l) - u_{cs,j}(k-l-1)] \right] \\
b_{ij}^-(k+1) &= f_w \left[b_{ij}^-(k) - \{x_i(k) - x_i(k-1)\} \sum_{l=1}^{\tau} \alpha_l b_{ij}^-(k-l) f_s[u_{cs,j}(k-l) - u_{cs,j}(k-l-1)] \right] \\
a_{ij}^+(k+1) &= f_w \left[a_{ij}^+(k) + \{x_i(k) - x_i(k-1)\} \sum_{l=1}^{\tau} \alpha_l a_{ij}^+(k-l) f_s[x_i(k-l) - x_j(k-l-1)] \right] \\
a_{ij}^-(k+1) &= f_w \left[a_{ij}^-(k) - \{x_i(k) - x_i(k-1)\} \sum_{l=1}^{\tau} \alpha_l a_{ij}^-(k-l) f_s[x_i(k-l) - x_j(k-l-1)] \right]
\end{aligned}$$

The functions $f_s(\cdot)$, $f_w(\cdot)$, and $f_w(\cdot)$ are shown in Figs. 2.2, 2.3, and 2.4, respectively. Note that $f_w(x) = -f_w(-x)$.

Single D-R Neuron

In the special case where the network consists of exactly one neuron, the network model is mathematically equivalent to the refined drive-reinforcement learning model for a single neuron, as described in [Klopf (1985)]:

$$\begin{aligned}
y(k) &= f_N[\{\mathbf{b}^+(k) + \mathbf{b}^-(k)\}^T \cdot \mathbf{u}_{cs}(k) + b^0 u_{us}(k)] \\
b_i^+(k+1) &= f_w \left[b_i^+(k) + \{y(k) - y(k-1)\} \sum_{l=1}^{\tau} \alpha_l b_i^+(k-l) f_s[u_{cs,i}(k-l) - u_{cs,i}(k-l-1)] \right] \\
b_i^-(k+1) &= f_w \left[b_i^-(k) - \{y(k) - y(k-1)\} \sum_{l=1}^{\tau} \alpha_l b_i^-(k-l) f_s[u_{cs,i}(k-l) - u_{cs,i}(k-l-1)] \right]
\end{aligned}$$

where it is assumed that the output of the neuron is simply its state [$c^0 = 1$], that there are no self-loops [$\alpha^+(k) = \alpha^-(k) = 0$], and that the neuron has only one unconditionable stimulus [$r_{us} = 1$]. With the following change of variables, these equations may be transformed into the equations governing the behavior of a single drive-reinforcement neuron (see [Klopf (1988)]):

$$\begin{aligned}
&\left. \begin{aligned} b_i^+(k) &\Rightarrow w_{2i-1}(k) \\ b_i^-(k) &\Rightarrow w_{2i}(k) \\ u_{cs,i}(k) &\Rightarrow x_i(k) \end{aligned} \right\} \quad \text{for } 1 \leq i \leq r_{cs} \\
&\left. \begin{aligned} b^0 &\Rightarrow w_{2i+1}(k) \\ u_{cs,i}(k) &\Rightarrow r_i(k) \end{aligned} \right\} \quad \text{for } i = r_{cs} + 1 \\
&\alpha_l \Rightarrow c_l \quad \text{for } 1 \leq l \leq \tau
\end{aligned}$$

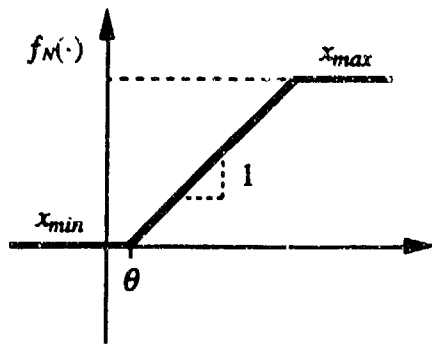


Fig. 2.1. Neuronal Output Function.

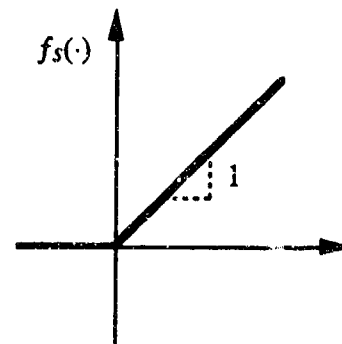


Fig. 2.2. Reinforcement Stimuli Function.

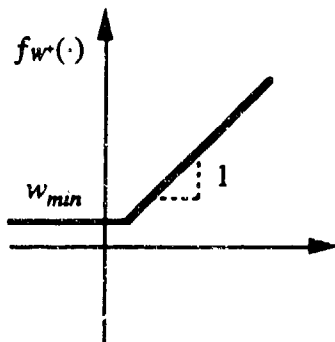


Fig. 2.3. Excitatory Adaptation Function.

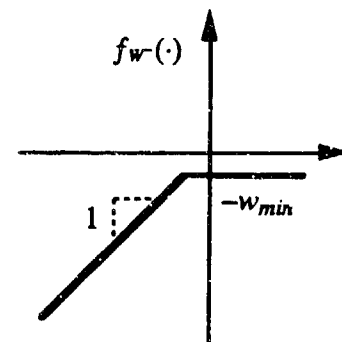


Fig. 2.4. Inhibitory Adaptation Function.

The network model of drive-reinforcement learning outlined above is quite general. Few restrictions have been made regarding the topology of the network; for example, self-loops have not been disallowed, nor has the influence of multiple unconditionable stimuli on the same neuron been ruled out. Such constraints might prove helpful in the refinement of the network model. If, for instance, drive-reinforcement neurons are not allowed to have self-loops, then each element on the main diagonal of the three matrices $A^+(k)$, $A^-(k)$, and A^0 , would necessarily have to be zero. Similarly, if a neuron is allowed to have at most a single unconditionable stimuli, then each row of the matrix B^0 , would have at most a single nonzero element.

2.1.2 Classical Conditioning Experiments

As mentioned above, a software simulation of these network equations was developed and implemented on a personal computer. The goal was to develop a tool with which we could easily explore the nuances of the algorithm. The special case of a network with only a single neuron was used to duplicate all of the classical conditioning experiments described in [Klopf (1988)]. The results we obtained were identical to those reported in this reference and, hence, will not be repeated below.

2.1.3 Analysis

In an attempt to better understand the D-R learning algorithm, we examined a number of its attributes, particularly in the special case of a single neuron. As a reminder, we provide the weight update equations for a single neuron below:

$$w_i^+(k+1) = f_w \left[w_i^+(k) + \Delta y(k) \sum_{j=1}^{\tau} \alpha_j |w_i^+(k-j)| f_s[\Delta u_i(k-j)] \right]$$
$$w_i^-(k+1) = f_w \left[w_i^-(k) + \Delta y(k) \sum_{j=1}^{\tau} \alpha_j |w_i^-(k-j)| f_s[\Delta u_i(k-j)] \right]$$

The equilibrium conditions of the input-output behavior of a single D-R neuron, as well as the equilibrium conditions associated with its adjustable weights, were examined under classical conditioning (open-loop) experiments. The conditions for zero weight change are shown below:

1. $w_i^+(k) = w_{min}$ and $\Delta y(k) < 0$
 $w_i^-(k) = -w_{min}$ and $\Delta y(k) > 0$
2. $\Delta y(k) = 0$
3. $\Delta u_i(k) \leq 0$ for $1 \leq j \leq \tau$

2.2 Further Work

After having gained a basic understanding of the drive-reinforcement learning paradigm under mostly open-loop conditions, we began to investigate its behavior

under closed-loop conditions. We were particularly interested in its convergence and stability properties under feedback.

Simple System Dynamics

Prior to the application of the D-R learning algorithm to the problem of controlling the cart-pole system, we elected to examine its performance relative to a simpler (two-dimensional) control problem with related dynamics. In particular, the "simple system" dynamics were:

- linear
- open-loop unstable
- nonminimum phase
- 2 state variables
- open-loop transfer function:

$$\frac{Y(s)}{U(s)} = \frac{(s - 3.8)}{s(s - 4.0)}$$

A variety of different controller configurations were explored, as were modifications to the basic D-R neuronal model to accommodate bipolar outputs. Initial work with this system showed promise, so we moved on to the full cart-pole control problem.

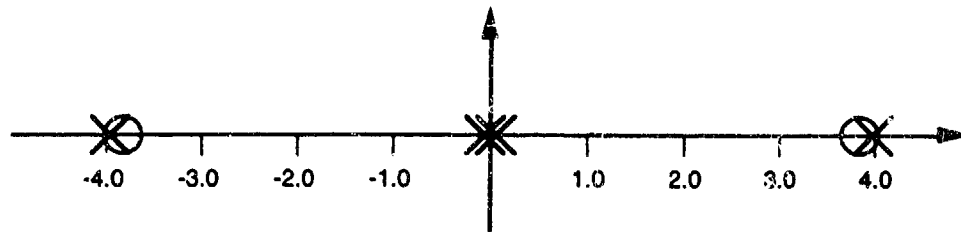
Cart-Pole System Dynamics

Nominal cart-pole system dynamics:

- nonlinear
- open-loop unstable
- nonminimum phase
- 4 state variables: $\{x, \theta, v, \omega\}$
- open-loop transfer function for linearized model:

$$\frac{X(s)}{F(s)} = \frac{(s - 3.8360)(s + 3.8360)}{s^2(s - 3.9739)(s + 3.9739)}$$

- open-loop poles and zeros in complex plane:



Under certain ideal conditions, a steady motion of the cart-pole system can be achieved:

- no noise or disturbances
- pole angle is constant and nonzero
- applied force is constant and nonzero
- pole angle, applied force, cart velocity, and cart acceleration all have same sign

Steady motion ensues if

$$\theta = \begin{cases} \tan^{-1} \left[\frac{f - \mu_c}{g(m_c + m_p)} \right] & \text{for } \theta, \dot{x} > 0 \text{ and } f > \mu_c \\ \tan^{-1} \left[\frac{f + \mu_c}{g(m_c + m_p)} \right] & \text{for } \theta, \dot{x} < 0 \text{ and } f < -\mu_c \end{cases}$$

The steady motion properties are important because they relate to pole-balancing experiments performed at Wright Laboratory.

WL Pole-Balancing Experiments

We were also fortunate enough to have access to a set of pole-balancing experiments (i.e., without restrictions on cart position) that were generated by researchers in the Avionics Directorate at the USAF Wright Laboratory. The basic scenario for this problem is outlined below:

- pole-balancing only
- infinite track
- failure if absolute value of pole angle exceeds 12 deg
- pole angle and angular rate information only
- sampling rate of 50 Hz
- response to initial 10 N disturbance from state-space origin

The WL pole-balancing results were successfully duplicated by our software simulation. However, in the course of this work, we obtained a number of interesting results that were not expected. These results were found, in part, as we sought to answer the following questions:

- what has the controller actually "learned"?
- is the acquired steady condition stable?
- what if the state-space quantization is refined?

Some of these issues are discussed in Section 2.3.

Chaotic Behavior

In the course of evaluating the WL pole-balancing results, we detected a discrepancy between software simulations run on different computers. One machine produced a very small round-off error (on the order of 10^{-18}) relative to the same (single) calculation performed on a second machine. Ordinarily, such a small numerical error is of no consequence; however, if the simulation is numerically unstable (as is the WL pole-balancing experiment, since cart position and velocity go to infinity), then even very small errors tend to grow to significant levels. This numerical instability combined with possible bifurcations due to different trajectories through the quantized state-space, results in extreme sensitivity to initial conditions and/or round-off errors (i.e., chaotic behavior). The upshot of this is that simulations of the WL pole-balancing experiment run on different machines can produce very different results.

Pole-Balancing Experiments Summary

Experiments performed:

- nominal
- perturbed+
- perturbed-
- random initial disturbances
- refined state-space quantization
- perturbed after steady motion achieved

Results:

- steady motion possible
- unstable dynamics
- small errors grow to significant levels
- trajectory bifurcations due to quantized state-space
- simulation is extremely sensitive to initial conditions and/or round-off errors

Elements of the solution:

- stable limit cycle about vertical must be attained
- control level in outer bins must provide sufficient restoring force
- cart position control may be subsequently achieved by biasing sensed pole angle

2.3 Refinement of D-R Network Equations

One way for a learning system to solve a control problem is by learning the appropriate feedback gains for the state variable and command inputs. Taken one step further, the network could also multiply these inputs by the gains it has learned, and then output the answer as the appropriate control action. Under these conditions, the input-output behavior of the network is essentially equivalent to that of a gain vector. The main difficulty encountered in applying a D-R learning network to this problem rests firmly with the fact that such a network is incapable of maintaining the feedback gains it has learned as the control system is exercised.¹ Alternatively, if the network attempts to learn the appropriate control actions directly (as a function of the state and input commands), the same difficulty remains.

Another way to state the basic underlying problem is as follows: in general, a D-R network that is used for feedback control will always be exposed to time-varying inputs (e.g., state variables, input commands, and performance measurements), and will always be expected to provide time-varying outputs (e.g., actuator commands), in accordance with the normal operation of the control system. In turn,

¹ The word "exercised" is used to describe the normal active use of a control system in which time-varying input commands are specified. For example, a control system might be used to position a cart-pole object on a flat horizontal track; if a time-varying command signal is used (e.g., the controller is told to move the cart-pole object back and forth between two fixed points on the track), then the control system is being *exercised*.

this implies that the neurons within such a network will also receive time-varying inputs and outputs. Under such conditions, the synaptic weights associated with these neurons will also vary as a direct consequence of the D-R learning algorithm. Thus, if one assumes that some D-R network has "learned" to behave as the desired controller at time t (which implies that it has acquired a suitable, but perhaps not unique, set of weights $w(t) = \{w_1, \dots, w_n\}$), then simply because any of the network inputs and outputs varies *in the course of normal operation* (i.e., under conditions in which no learning is required), *some network weights will change* as well. As a result, the new network weights will no longer correspond to the set of weights associated with the controller that previously had "learned" the desired control law (i.e., $w(t) \neq w(t+1)$, in general). Our conclusion is that a network of D-R learning neurons (as narrowly defined in [Klopf (1988)]) suffers from an instability problem when used under closed-loop conditions, in the sense that the weights do not appear to have any stable equilibrium configurations when the closed-loop system is exercised. No "reasonable" network structures have been identified which overcome this problem.

2.3.1 Multiplication Learning Problem

A very simple closed-loop control problem that can be used to illustrate some of the foregoing concepts is based on a network that takes one input and learns to provide a single output that is the input times a constant gain k . This highly simplified control problem will be called the *multiplication learning problem*.

To help the network learn the desired gain k , it will have access to feedback signals that give it clues concerning its current behavior. In this problem, we assume that the network has access to a single feedback signal that tells it whether its output is too high or too low, and by how much. This signal will be called the error signal.

Clearly, for some learning algorithms this problem can be solved by a network comprised of a single plastic weight: the input is multiplied by the weight to give the output, and the weight is adjusted proportional to the error signal. In fact, a weight update algorithm can be easily derived that allows "deadbeat" control (learning occurs in a single time step). Performance at this level will not be required, however; it will be acceptable if the plastic weight approaches k asymptotically.

ically, as it would if a gradient learning algorithm were used. It would even be acceptable if the weight fluctuated around the correct value, as long as it stayed "in its neighborhood." Thus, the problem constraints have been relaxed somewhat by requiring that the network *eventually* learn to maintain its output within some prescribed range of the desired output.

Some potential approaches for solving the multiplication learning problem involve feedback loops within the network. In such cases, it may take a significant amount of time for an output to be generated by a given input. To make the problem easier for the learning network, it will also be assumed that the input only changes at periodic intervals, and that the output associated with each input must be calculated before the end of an input interval. The network designer in this problem is free to create a network of any finite size, with any finite number of plastic weights, and may also assume that the input changes at any desired finite periodic rate. These rules give the designer a great deal of freedom, and should make the problem as easy as is possible without radically changing its nature.

This problem is a simplified version of what a realistic control system should be able to accommodate and is, therefore, probably something that *any* learning control system should be able to accommodate. D-R networks appear to be unable to solve this problem, even if large hierarchical networks are used. In general, this is because the solution k must be stored in a plastic weight somewhere in the network, and must periodically have the input signal applied to it. Since the input signal may change several times within one τ period, input-output correlations will occur that cause the weight to change, even when it is already at the correct value of k . Stable equilibrium weight configurations are essentially unobtainable in a D-R learning network subject to closed-loop conditions.

2.3.2 Facilitatory Learning

Facilitatory learning (and/or other ancillary learning mechanisms) may complement the basic D-R learning algorithm in such a way as to allow D-R learning to be successfully applied to feedback control problems. We have some (untested) ideas on this subject.

Weight update equations for a single modified neuron:

$$w_i^+(k+1) = f_w \left[w_i^+(k) + \Delta u_i^m(k) \sum_{j=1}^{\tau} \alpha_j |w_i^+(k-j)| f_d[\Delta u_i(k-j)] \right]$$

$$w_i^-(k+1) = f_w \left[w_i^-(k) + \Delta u_i^m(k) \sum_{j=1}^{\tau} \alpha_j |w_i^-(k-j)| f_d[\Delta u_i(k-j)] \right]$$

- weight change mediated by an otherwise inert stimulus
- self-loop (without delay ...) corresponds to unmodified D-R neuron

These proposed modifications are similar in effect to subsequent refinements of the drive-reinforcement paradigm made by its developers, particularly in the special case where it is embedded in an associative control process network (e.g., see [Klopf, Morgan, Weaver (1992)]). As we later discovered, this refinement does allow the combined D-R/ACP approach to be successfully applied to optimal control problems—this work is fully described in Attachment 3.

2.4 Alternatives Strategies for Learning Control

In preparation for the comparative evaluation phase of the program, we considered alternative approaches, including conventional adaptive control and Barto-Sutton learning control. In addition, we explored key issues related to control system configuration and control system performance. These issues have not been fully resolved.

2.4.1 Performance Measures & Design Issues

The key performance measures and design issues that we considered in our evaluation are summarized below.

Performance measures:

- stability
- transient response
 - settling time
 - overshoot
- steady-state behavior
 - tracking error
 - limit cycles

- robustness to modeling uncertainty
- sensitivity to noise and disturbances
- control action (energy, power)
- adaptation time
- optimality (local vs. global)

Other issues:

- controller design difficulty
 - structure
 - parameters
 - training process
- information requirements
 - *a priori* (design and training)
 - real-time (measurements)
- implementation
 - processing requirements
 - storage requirements
 - sequential vs. parallel
 - cycle time
- scale-up to more difficult problems

2.4.2 Alternative Approaches

A number of alternative approaches to drive-reinforcement learning were considered as candidates for subsequent application to the flight control problem. Several conventional approaches were included in this comparison to provide a stronger basis for comparison. The results of this comparison are summarized below.

Linear State Feedback Control

Basis:

- linear combination of the state variables
- [Kailath (1980); Maybeck (1979)]

Advantages:

- well-developed theory

- "turnkey" design process
- wide-ranging applicability

Disadvantages:

- linear and quasi-linear systems only
- sensitive to modeling uncertainty
- not adaptive

Gain Scheduling

Basis:

- multiple linear controllers
- switching or interpolation
- [Åström & Wittenmark (1989)]

Advantages:

- nonlinear control law

Disadvantages:

- extensive manual tuning required
- *ad hoc* design \Rightarrow "black art"
- not adaptive

Indirect Adaptive Control

Basis:

- parameter estimation (e.g., recursive least squares)
- linear design (e.g., pole-placement, LQR)
- [Gupta (1986); Narendra, Ortega, & Dorato (1991)]

Advantages:

- adaptive
- flexible design

Disadvantages:

- age-weighting or resetting required
- persistent excitation required
- identification problems

Model Reference (Direct) Adaptive Control

Basis:

- explicit description of desired system behavior (reference)
- gradient algorithm
- [Åström & Wittenmark (1989); Narendra & Annaswamy (1989)]

Advantages:

- identification not required

Disadvantages:

- stability problems for nonminimum phase systems
- local optimization only

Direct Adaptive Control: TDC

Basis:

- "time-delay" control: adaptive nonlinear transformation
- [Youcef-Toumi & Ito (1990)]

Advantages:

- simple algorithm
- exceptional flexibility

Disadvantages:

- sensitive to "B" matrix uncertainty (e.g., unknown actuator dynamics)
- high sampling rate required
- state derivative required

ASE/ACE Learning Control

Basis:

- associative reinforcement learning
- [Barto, Sutton, & Anderson (1983)]

Advantages:

- simple learning algorithm
- optimizing

Disadvantages:

- bang-bang control laws only
- state-space quantization required

- high storage requirements
- performance does not improve with refinement of quantization scheme
- difficult to specify control objective

None of the various approaches considered above was completely satisfactory for the objectives we had in mind. In addition to carrying out software simulations of the basic algorithms, we also conceived and implemented a number of modifications to these algorithms (e.g., a type of "annealing" algorithm for the ASE/ACE paradigm). Even so, none appeared to be suitable for our application to flight control. However, as a result of the experience and insight gained during our examination of these various approaches as well as the D-R learning paradigm, we were able to first conceive and then develop a novel hybrid adaptive/learning control approach that was ultimately successful. This approach is the subject of most of the remaining chapters.

2.5 D-R Revisited: ACP Networks & Learning for Optimal Control

As mentioned at the outset, a decision was made to revisit the drive-reinforcement learning paradigm in the special case where it is embedded in an associative control process network. The results of this phase of the program are fully described in Attachment 3.

3 Learning for Flight Control

This chapter provides a high-level discussion of the motivation for, as well as the issues underlying, the use of learning in flight control applications.

After a brief introduction in Section 3.1, some background material on the topic of flight control system design is presented in Section 3.2. Key differences between adaptation and learning (in this context) are outlined in Section 3.3. Section 3.4 motivates and presents a high-level description of the hybrid adaptive/learning control methodology, which will be developed in more detail in the next chapter. Section 3.5 elaborates on the idea that learning (in the context of control) can be considered as automatic (on-line) function synthesis. Section 3.6 discusses a number of important application issues associated with the use of adaptation and learning in flight control. The potential benefits of learning augmentation are then summarized in Section 3.7.

3.1 Introduction

The design of automatic control systems for high performance aircraft represents a difficult and challenging problem because of the coupled, multivariable, nonlinear, and time-varying nature of flight dynamics, in conjunction with the uncertainties associated with existing aerodynamic vehicle models. The added specification of "high performance" generally implies an expanded flight envelope and faster dynamics--attributes that only exacerbate the problem. Conventional control system design methods for such systems have a number of important limitations. Fixed parameter, off-line control system design approaches (e.g., based on gain scheduling, dynamic inversion, or extended linearization) are suitable for nonlinear problems where there is little or no model uncertainty; in practice, they often require extensive manual tuning (which can lead to excessive development costs) because the physical models used during the design process do not always accurately reflect the actual system dynamics. Robust design methods deal with the problem of model uncertainty, but may sacrifice closed-loop system performance as a result, and may be impractical for problems involving significant nonlinearity or time-varying dynamics. Adaptive control approaches can accommodate some parametric model uncertainty and slowly time-varying dynam-

ics, but may be unsuitable or inefficient for problems involving significant structural model uncertainty (e.g., significant nonlinearity).

An alternative approach relies on the use of *learning* to augment the flight control system. This approach can directly accommodate parametric uncertainty and some structural uncertainty (including memoryless nonlinearities). Learning systems offer unique capabilities that may be exploited to provide superior flight control systems. The approach we have pursued relies on special-purpose connectionist systems that can be used for *on-line* learning. The key concept underlying our methodology is the view that

learning may be interpreted as the automatic synthesis of multivariable functional mappings, based on experiential information that is gained incrementally over time, and a criterion for optimality.

When combined with adaptation, the resulting *hybrid control* strategy provides a powerful control system design and implementation technique. For flight control applications, we argue that advanced control systems incorporating learning may be used advantageously to:

- *facilitate the control system design and tuning process*
- *accommodate modeling error through on-line interaction with the actual vehicle*
- *improve performance through on-line self-optimization*
- *improve efficiency by reducing undesirable transient effects that would ordinarily be induced by parameter adjustment in a purely adaptive controller*

In a fundamental sense, the flight control system design problem is to find an appropriate *functional mapping*, from measured and desired vehicle outputs, to a set of control actions that will produce satisfactory behavior in the closed-loop system. In other words, the problem is to choose a function (a *control law*) that achieves certain closed-loop performance objectives when applied to the open-loop system. In turn, the solution to this problem may naturally involve other mappings (recall Section 1.1). In general, these mappings may represent static or dynamic functions.

If there is adequate design information (i.e., if all pertinent vehicle dynamic and aerodynamic models are available) and if there is little or no significant uncertainty in this data, then (in principal) the mappings required to produce a satisfactory flight control system can be designed and developed through a completely off-line process, which results in an *a priori* controller. Unfortunately, this situation rarely exists in practice, particularly when the design is for a system as complex as a modern high performance aircraft. At the very least, manual tuning of the nominal control law will be required, following initial flight testing.

The need for learning, in the context of flight control, arises in situations where a system must operate in *conditions of uncertainty*, and where the available *a priori* information is so limited that it is impossible or impractical to design in advance a system that has fixed properties and also performs sufficiently well [Tsypkin (1973)]. Current design trends for high performance aircraft (e.g., flight envelope expansion into more complex and less understood flight regimes) suggest that the *traditional* off-line design approach, followed by flight test and manual tuning, will become increasingly difficult, perhaps even to the point where this approach is no longer adequate, nor cost-effective [Baker & Farrell (1991); Steinberg (1992)]. Accordingly, a central role of learning in flight control is to enable a wider class of problems to be solved, by reducing the prior uncertainty to the point where satisfactory solutions can be obtained, in part, on-line.

3.2 Background: Flight Control System Design

Subsection 3.2.1 briefly reviews some of the difficulties that may be encountered in the design of flight control systems, while Subsection 3.2.2 describes limitations of the traditional approaches that are used to address these problems. Issues that are not related to learning augmentation per se have been purposely excluded from the discussion.

3.2.1 Design Difficulties

An effective flight control system design must address several difficulties related to the complex dynamical behavior of aircraft, as well as a further difficulty arising from modeling errors.

Multivariable Control & Dynamic Coupling

Because aircraft rely on multiple effectors (e.g., ailerons, canards, elevators, rudders, and thrust-vectoring) to simultaneously control a number of outputs (e.g., attitude and attitude rates), the control system design problem is formally a multivariable one. Due to a combination of rigid-body and aerodynamic effects, the principal state variables associated with aircraft flight dynamics (altitude, velocity vector, orientation angles, and angular velocity vector) are coupled via the equations-of-motion. Thus, for example, nonzero roll rates can cause yaw rate changes (i.e., roll-yaw coupling). In some cases, it may be possible to decouple such input/output modes and design multiple independent single-input/single-output (SISO) controllers. However, this approach has the disadvantage that system performance will often be sacrificed, since, in general, a multivariable control system may be necessary to fully exploit the dynamical potential of the vehicle and obtain maximal system performance (e.g., maximal maneuvering capability). The design of a full multivariable control system is often considerably more difficult than the design of multiple independent SISO controllers, due to the higher dimensionality and dynamic coupling associated with the system.

Nonlinearity

The dynamical behavior of all aircraft is inherently nonlinear. This is due to: (i) aerodynamic forces and moments that are complex nonlinear functions of aircraft state, (ii) dynamic coupling terms that have a nonlinear form, (iii) actuators and effectors that have physical limitations (e.g., saturation and rate limits), and (iv) nonlinear engine dynamics. For example, control surface effectiveness depends on the speed, altitude, and attitude of the aircraft—under some conditions a control surface may become ineffective (e.g., in a stall) or even reverse its effectiveness (e.g., as in aileron reversal). Standard aircraft equations-of-motion are structurally nonlinear, moreover, the "parameters" used are often themselves nonlinear functions of the aircraft state. Under such conditions, a single fixed gain linear control design will generally be inadequate.

Time-Varying Dynamics

Additional control system design difficulties arise because the dynamical behavior of an aircraft can change over time. The *effect* of these variations may be either predictable or unpredictable. Sources of time-varying dynamics whose effect cannot be predicted include disturbances (e.g., wind gusts), component degradation, and component failures. In contrast to these sources of time-varying dynamics, there are others whose effect can be predicted. For instance, aircraft dynamical behavior varies (in a predictable manner) as a function of configuration changes (e.g., wing sweep), fuel use, or payload deployment. These particular time-varying behaviors can actually be construed as spatial dependencies, where the state of the vehicle is augmented to include variables (which may be analog or discrete-valued) such as WING_SWEEP_ANGLE, REMAINING_FUEL, or PAYLOAD_DEPLOYED. To accommodate such predictable temporal variations, a control system must be explicitly designed to do so (e.g., via gain scheduling), be adaptive, or be robust to such effects.

Model Uncertainty

Aerodynamic vehicle models are susceptible to two types of uncertainty: structural and parametric. Structural uncertainty arises when the assumed mathematical form of the equations-of-motion (e.g., the standard six-degree-of-freedom aircraft model) is unable to adequately describe the behavior of the vehicle throughout the operating envelope. This means that no fixed (constant) set of globally correct model parameters exists. To account for this, the parameters in most aircraft models are scheduled as a function of other variables. In turn, this generally implies that the functional relationship between these scheduling variables and the model parameters is not known in closed-form (a second possibility is that the relationship is too complex to be represented conveniently in closed-form). Even if the presumed equations-of-motion were capable of accounting for all aerodynamic forces and moments (i.e., even if there were no structural uncertainty), empirical errors incurred during the estimation of the model parameters (e.g., due to measurement noise) would still result in some parametric uncertainty and, hence, discrepancies between the actual and simulated vehicle behavior. In general, model-based control system design can only be as good as the un-

derlying model, and if there is significant uncertainty in this model, the results may be catastrophic.

Design Trends

As high performance aircraft continue to evolve, the desired flight envelope is likely to expand; indeed, the general trend is towards new flight regimes that are, at the same time, more complex and less understood (e.g., post-stall maneuvering). Trends that are likely to exacerbate the design problem include:

- flight envelope expansion \Rightarrow increasingly nonlinear and unknown regimes
- additional control effectors (e.g., vectored thrust) \Rightarrow higher dimensionality
- relaxed-static-stability and agility \Rightarrow faster control response needed

In the future, flight control system design may become increasingly difficult, perhaps even to the point where traditional methods are no longer adequate, nor cost-effective.

3.2.2 Traditional Design Approaches

Three different approaches to flight control system design are discussed below, in the context of the design difficulties outlined in the previous subsection. In a general sense, the use of learning does not preclude the use of these other techniques; in fact, the main advantages of learning are realized by using it (in an appropriate manner) to augment existing control methodologies.

Robust Control

Robust control design methods attempt to explicitly incorporate robustness to parametric and structural model uncertainty into the control system design [Maciejowski (1989)]. In the *ideal* case where the design model is perfect and there is no uncertainty, maximal closed-loop system performance can be obtained through an appropriate optimal feedback control law. Acknowledging that some level of model uncertainty exists, *robust control* techniques (e.g., H_∞ , μ -synthesis, or even classical gain and phase margin based approaches) can be used to design a fixed parameter control system that will provide a guaranteed level of (sub-maximal) performance for *any* plant in a prescribed set of likely plants. This set might, for instance, be described by a simplified model (typically linear) and a

bounded set of nominal model parameters (usually, a physically realistic range is specified for each model parameter). The parameter ranges must be large enough to ensure that the resulting set of models contains the behavior of the actual plant. The robust control system design problem is essentially a minimax optimization problem; its solution tends to be conservative in the sense that the *best* that can be achieved is often dictated by the *worst case* scenario. Thus, a tradeoff exists between performance and robustness, and robust control designs are achieved at the expense of resulting closed-loop system performance—relative to a control design based on a perfect model.

To some extent, robustness to nonlinear and time-varying dynamical behavior may also be obtained through robust design methods, since these effects can be considered to be a component of the model uncertainty, and can therefore be accommodated by increasing the level of the prescribed model uncertainty. The disadvantage of this approach is that it can result in very conservative assumptions about the level of uncertainty and, thus, may lead to even lower levels of overall system performance. There is probably sufficient uncertainty in the standard aircraft equations-of-motion so that high closed-loop system performance can only be obtained in one of two ways: (i) through extensive manual (off-line) tuning of the nominal control law design, based on flight test data or (ii) via an automatic on-line adjustment technique—a fixed *a priori* robust control design will not necessarily suffice.

Gain Scheduling / Manual Tuning

A traditional control system design methodology for high performance aircraft is based on *gain scheduling* [Åström & Wittenmark (1989); Kreisselmeier (1986); Stein (1986)]. In this scheme, multiple linear controllers are used to approximate the required nonlinear control law. A separate linear controller must be designed for each member of an *ad hoc* set of distinct regions that together cover the complete flight envelope. In each region the dynamical behavior of the aircraft is assumed to be linear (multiple regions are required since the aircraft dynamics cannot be accurately represented by a single linear model). Nonlinear dynamical effects are addressed by transitioning between locally applicable linear controllers. The complete nonlinear control law is realized by interpolating between these separate linear controllers in a preprogrammed manner, as a function of

the current state of the vehicle. The number of distinct locally linear operating points that might be considered can range into the hundreds.

The *ad hoc* and localized nature of the gain scheduling design approach can result in numerous design iterations, each involving manual redesign of the nominal control law (for certain flight conditions), followed by extensive computer simulation to evaluate the modified control law. After the initial control system has been designed and validated in simulation, further difficulties may arise because the models used during the design process will not always accurately reflect the actual vehicle dynamics. Extensive on-line tuning of the nominal control system may be required to achieve satisfactory performance. Moreover, since the feedback gains are scheduled in an open-loop fashion, no automatic corrective action is taken to mitigate the effects of a control law that is no longer appropriate. Hence, time-varying dynamics and other unanticipated events (e.g., performance degradation and changes in the vehicle configuration or environment) are only indirectly addressed through the limited robustness of the final control system design. In the future, conventional gain scheduled (and manually tuned) control system design methods may become increasingly difficult as a direct result of the growing complexity and sophistication of new high performance aircraft.

Adaptive Control

Many different adaptive control methods for nonlinear and time-varying systems have been investigated [Åström & Wittenmark (1989); Gupta (1986); Narendra & Annaswamy (1989); Narendra & Monopoli (1980); Narendra, Ortega, & Dorato (1991); Slotine & Li (1991)]. Two generic strategies are briefly described here. *Indirect adaptive control* approaches utilize an explicit dynamical model of the vehicle, which is updated periodically, to synthesize new control laws. This approach has the advantage that powerful design methods (including optimal control design techniques) can be employed on-line; however, it has the disadvantage that on-line model identification is required. Alternatively, *direct adaptive control* approaches do not rely upon a vehicle model and, thus, avoid the need to perform explicit on-line model identification. Instead, the control law is adjusted directly, based on the observed dynamical behavior of the vehicle. In either case, the control system will attempt to adapt if the behavior of the vehicle changes by a significant degree.

Adaptive control systems are dynamic systems and require finite time intervals to properly detect and account for variations in the vehicle or its environment. If the dynamical characteristics of the vehicle vary considerably over its operating envelope (e.g., due to nonlinearity), then the control system may be adapting most of the time (i.e., it may always be in a "partially" adapted state), resulting in degraded performance. Note that this can occur even in the absence of time-varying dynamics and disturbances, since the control system must readapt *every* time a different dynamical regime is encountered (i.e., one that is outside the scope of the current control law). These issues are particularly relevant to flight control applications, where vehicle behavior is strongly dependent upon flight condition. For these and other reasons, most control system designs for high performance aircraft have been based on gain scheduling, rather than on adaptive methods [Kreisselmeier (1986); Stein (1986)]. As will be discussed in the next section, learning systems may be used to simultaneously overcome some of the shortcomings and complement many of the advantages of adaptive control systems.

3.3 Adaptation vs. Learning

In addition to *modeled* (i.e., known) nonlinearities or time-varying dynamics, an effective automatic control system must overcome difficulties arising from two sources: (i) noise, disturbances, and unmodeled time-varying dynamics and (ii) unmodeled nonlinearities, dynamic coupling, and other spatial dependencies. The first has a *temporal* emphasis and represents dynamical features that are essentially unpredictable; in contrast, the second source of difficulty has a *spatial* emphasis and represents dynamical features that are predictable. For example, an advanced flight control system for a high performance aircraft could eventually "learn" to anticipate the nonlinear aspects of the vehicle behavior, but could never anticipate noise, disturbances, or unmodeled time-varying dynamics.

In much the same way that adaptive approaches can be considered as an extension of fixed parameter (nonadaptive) control methods, where on-line adjustment of control system parameters is used to compensate for simple model uncertainty, learning approaches can be considered as an extension of adaptive control methods, where on-line synthesis of functional relationships is used to accommodate more complex model uncertainty. Adaptive methods can be used for applications involving unknown (but constant or slowly time-varying) model parameters like

vehicle inertial properties, while learning methods can be used for unknown (but quasi-static) spatial dependencies like control surface effectiveness as a function of angle-of-attack. Both methods utilize experiential information gained through closed-loop interactions with the vehicle and environment to improve their performance during subsequent interactions.

The key differences between adaptation and learning are essentially a matter of degree and emphasis. Adaptive control has a temporal emphasis: its objective is to maintain desired closed-loop behavior in the face of disturbances and dynamics *that appear to be time-varying*. In actuality, the changing dynamics may be caused by unmodeled nonlinear effects, so that they are really a function of state rather than of time. Because the functional form of most adaptive control laws is generally incapable of representing, over a wide range of operating conditions, the required control action as a function of the current vehicle state, it can be said that adaptive controllers lack "memory" in the sense that they must readapt to compensate for all changing dynamics, *even those which are nonlinear (but time-invariant) and have been experienced previously*. This inefficiency can result in degraded performance, since transient behavior due to parameter adjustment may occur every time the presumed dynamical behavior of the vehicle changes by a sufficient degree.

In general, adaptive controllers lack the ability to distinguish between temporally and spatially dependent variations in the dynamics of a vehicle. They operate, in effect, by optimizing a small set of adjustable parameters to account for vehicle behavior that is local in both space and time. To be effective, adaptive controllers must have relatively fast dynamics so that they can quickly react to changing vehicle behavior.

Learning augmented control strategies differ from those of conventional adaptive control primarily with respect to "memory" (in the same sense as used above), use of past information, and emphasis. The contrast between adaptation and learning is particularly relevant to the control of high performance aircraft. Learning augmented controllers exploit an automatic mechanism that *associates*, throughout some operating envelope, a suitable control action or set of control system parameters with the current flight condition. In this way, the presence and effect of previously unknown nonlinearities can be anticipated and ac-

counted for, based on past experience. Once such a control system has "learned," transient behavior that would otherwise be induced by parameter adaptation no longer occurs, resulting in greater efficiency and improved performance over adaptive control strategies. To accomplish this, *learning control systems rely on general function approximation schemes* that may be used, for example, to map the current flight condition to an appropriate set of control system parameters (in this regard, the net effect of learning is similar to that of gain scheduling, with the proviso that learning has occurred *on-line* with the *actual vehicle*, while gain scheduling is developed *off-line* via a *model*).

Learning control has a spatial emphasis. For example, its objective might be to synthesize a feedback control law (as a function of vehicle state) that provides the desired closed-loop behavior in the presence of unmodeled nonlinear dynamics. Alternatively, learning can be used to synthesize a mapping from flight condition to a set of linear model parameters, which can then be used for on-line control law design.¹ Learning systems operate by optimizing over a large set of adjustable parameters (and potentially variable structural elements [Cerrato (1993)]) to construct a mapping representing the quasi-static spatial dependencies, again, throughout the operating envelope. In effect, this optimization is global in state-space. To successfully execute this optimization process, learning systems make extensive use of past information and employ relatively slow learning dynamics.

As defined, the processes of adaptation and learning are complementary: each has unique desirable characteristics from the point of view of flight control. For example, adaptive approaches address the problem of slowly time-varying dynamics and novel situations (e.g., those which have never before been experienced), but are inefficient for problems involving significant unknown spatial dependencies. Learning approaches, in contrast, have the opposite characteristic: they are well-equipped to accommodate nonlinear vehicle dynamics, but are not well-suited to applications involving time-varying dynamics.

¹ This second approach is the learning analog to *indirect* adaptive control, whereas the first approach is analogous to *direct* adaptive control; cf. [Atkins (1993)].

3.4 Motivation for Hybrid Adaptive/Learning Control

This section briefly describes hybrid control system architectures that exhibit both adaptive and learning behaviors. These hybrid structures incorporate adaptation and learning in a synergistic manner. In such schemes, an adaptive system is coupled with a learning system to provide real-time adaptation to novel situations and slowly time-varying dynamics, in conjunction with learning to accommodate stationary or quasi-stationary state-space dependencies (e.g., memoryless nonlinearities). The adaptive control system reacts to discrepancies between the desired and observed behaviors of the plant, to maintain the requisite closed-loop system performance. These discrepancies may arise from time-varying dynamics, disturbances, or unmodeled dynamics. In practice, little can be done to anticipate time-varying dynamics and disturbances; thus, these phenomena are usually handled through feedback in the adaptive system. In contrast, the effects of some unmodeled dynamics (in particular, static nonlinearities) can be predicted from previous experience. This is the task given to the learning system. Initially, all unmodeled behavior is handled by the adaptive system; eventually, however, the learning system is able to *anticipate* previously experienced, yet initially unmodeled behavior. Thus, the adaptive system can concentrate on novel situations (where little or no learning has occurred) and slowly time-varying behavior.

Two general hybrid architectures are outlined in this section. The discussion of these architectures parallels the usual presentation of direct and indirect adaptive control strategies. In each approach, the learning system is used to alleviate the burden on the adaptive controller of continually reacting to predictable state-space dependencies in the dynamical behavior of the plant (e.g., stationary, memoryless nonlinearities). Note that various technical issues must be addressed to guarantee the successful implementation of these approaches. For example, to ensure both the stability and robustness of the closed-loop system (which includes both the adaptive and learning systems, as well as the plant), one must address issues related to: controllability and observability, the effects of noise, disturbances, model-order errors, and other uncertainties; parameter convergence, sufficiency of excitation, and nonstationarity; computational requirements, time-delays, and the effects of finite precision arithmetic. Many (if not all) of these issues arise in the implementation of traditional adaptive control systems; as such, there are some existing sources one may refer to in the hope of addressing these

issues (e.g., see [Åström & Wittenmark (1989); Gupta (1986); Narendra & Annaswamy (1989); Narendra & Monopoli (1980); Narendra, Ortega, & Dorato (1991); Slotine & Li (1991)]). Although these topics are well beyond the scope of this report, in some instances the learning augmented approach appears to offer operational advantages over the corresponding adaptive approach (with respect to such implementation issues).

3.4.1 Direct Implementation

In the typical *direct* adaptive control approach (see Fig. 3.1), each control action u is generated based on the measured y_m and desired y_d plant outputs, internal state of the controller, and estimates of the pertinent control law parameters \mathbf{k} . The estimates of the control law parameters are adjusted, at each time-step, based on the error e between the measured plant outputs and the outputs of a reference system y_r . Of course, care must be taken to ensure that the plant is actually capable of attaining the performance specified by the selected reference system. Direct adaptive control approaches do not rely upon an explicit plant model and, thus, avoid the need to perform on-line system identification.

The controller in Fig. 3.1 is structured so that normal adaptive operation would result if the learning system were not implemented. The reference represents the desired behavior for the augmented plant (controller plus plant), while the adaptive mechanism is used to transform the reference error directly into a correction $\Delta \mathbf{k}$ for the current control system parameters. The adaptation algorithm can be developed and implemented in several different ways (e.g., via gradient or Lyapunov based techniques—see [Åström & Wittenmark (1989); Narendra & Annaswamy (1989); Slotine & Li (1991)]). Learning augmentation can be accomplished by using the learning system to store the required control system parameters as a function of the operating condition of the plant [Farrell & Baker (1992); Vos, Baker, & Millington (1991)]. Alternatively, learning can be used to store the appropriate control action as a function of the actual and desired plant outputs [Farrell & Baker (1991)]. The architecture in Fig. 3.1 shows the first case.

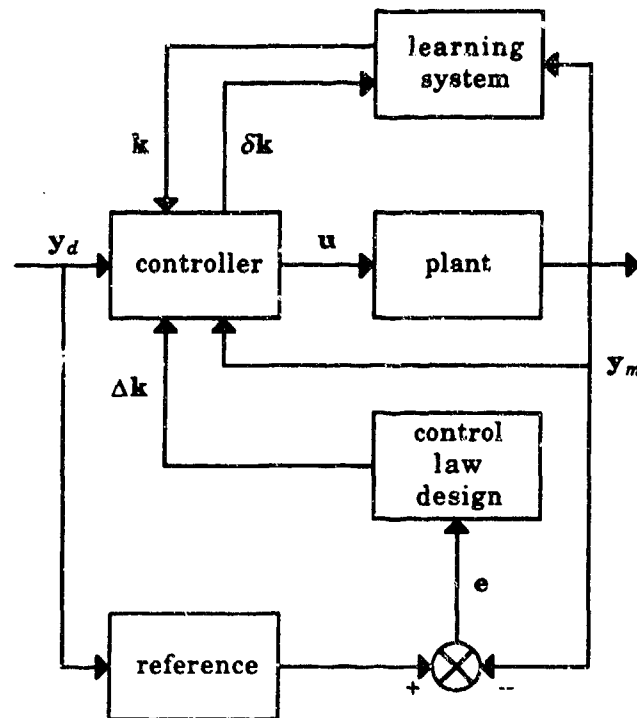


Figure 3.1. Direct Adaptive/Learning Approach.

When the learning system is used to store the control system parameters as a function of the plant operating condition, the adaptive system would provide any required *perturbation* to the control parameters \mathbf{k} generated by the learning system. The signal from the control block to the learning system in Fig. 3.1 is the perturbation in the control parameters $\delta\mathbf{k}$ to be associated with the previous operating condition. This association (incremental learning) process is used to combine the estimate from the adaptive system with the control parameters that have already been learned for that operating condition. At each sampling instant, the learning system generates an estimate of the control system parameters \mathbf{k} associated with that operating condition, and then passes this estimate to the controller where it is combined with the perturbation parameter estimates maintained by the adaptive system, and used to generate the control action \mathbf{u} . In the ideal limit where perfect learning has occurred, and there is an absence of noise, disturbances, and time-varying dynamics, the correct parameter values would always be supplied by the learning system, so that both the perturbations $\delta\mathbf{k}$ and

corrections $\Delta \mathbf{k}$ generated by the adaptive system would become zero.¹ Under more realistic assumptions, there would be some small degradation in performance due to adaptation (e.g., $\delta \mathbf{k}$ and $\Delta \mathbf{k}$ might not be zero due to noise).

In the case where the learning system is trained to store control action directly as a function of the actual and desired operating conditions of the plant, the adaptive system would provide any required perturbation to the control action generated by the learning system. Note that a dynamic mapping would have to be synthesized by the learning system if a dynamic feedback law were desired (which was not necessary in the first case). The advantage of this approach over the previous one is that a more general control law can be learned. The disadvantage is that additional memory is required and that a more difficult learning problem must be addressed.

3.4.2 Indirect Implementation

In the typical *indirect* adaptive control approach (see Fig. 3.2), each control action u is generated based on the measured y_m and y_d desired plant outputs, internal state of the controller, and estimated parameters $\hat{\mathbf{p}}_a$ of a local plant model. The parameters \mathbf{k} for a local control law are explicitly designed on-line, based on the observed plant behavior. If the behavior of the plant changes (e.g., due to nonlinearity), an estimator automatically updates its model of the plant as quickly as possible, based on the information available from the (generally noisy) output measurements. The indirect approach has the important advantage that powerful design methods (including optimal control techniques) may potentially be used on-line. Note, however, that computational requirements are usually greater for indirect approaches since both model identification and control law design are performed on-line.

If the learning system in Fig. 3.2 were not implemented, then this structure would represent the operation of a traditional indirect adaptive control system. The signal $\hat{\mathbf{p}}_a$ is the adaptive estimate of the plant model parameters. This signal

¹ In this case, the system architecture is similar to that used in gain scheduling, with the proviso that learning has occurred on-line with the *actual plant*, while a gain schedule is developed off-line via a *model*.

is used to calculate the control law parameters \mathbf{k} . Incorporation of the learning system would allow the plant model parameters to be learned as a function of the plant operating condition. The model parameters generated by the learning system allow previously experienced plant behavior to be *anticipated*, leading to improved control law design [Baird & Baker (1990)]. In this case, the output of the learning system \mathbf{p}_l to both the control design block and the estimator is an *a priori* estimate of the model parameters associated with the current operating condition. An *a posteriori* parameter estimate \mathbf{p}_{post} from the estimator (involving both filtering and posterior smoothing) is used to update the mapping stored by the learning system. The system uses model parameter estimates from both the adaptive and learning systems to execute the control law design and determine the appropriate control law parameters. In situations where the design procedure is complex and time-consuming, the control law parameters might also be stored (via a separate mapping in the learning system) as a function of the plant operating condition. Thus, control law design could be performed at a lower rate, assuming that the control parameter mapping maintained by the learning system was sufficiently accurate to provide reasonable control in lieu of design at a higher rate.

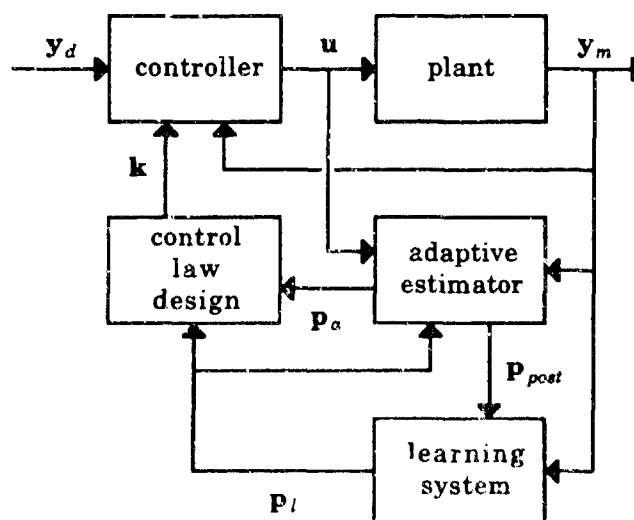


Figure 3.2. Indirect Adaptive/Learning Approach.

3.4.3 Summary of Hybrid Control Architectures

In both of the hybrid implementations described in this section, the learning system (prior to any on-line interaction) would only contain knowledge derived from the design model. During initial closed-loop operation, the adaptive system would be used to accommodate any inadequacies in the *a priori* design knowledge. Subsequently, as experience with the actual plant was accumulated, the learning system would be used to anticipate the appropriate control or model parameters as a function of the current plant operating condition. The adaptive system would remain active to handle novel situations and limitations of the learning system (e.g., finite accuracy). With perfect learning, but no noise, disturbances, or time-varying behavior in the plant, the contribution from the adaptive system would eventually become zero. In the presence of noise and disturbances, the contribution from the adaptive system would become small, but nonzero (depending on the hybrid scheme used, however, the *effect* of this contribution might be negligible). In the general case involving all of these effects, the hybrid control system should perform better than either subsystem individually. It can be seen that adaptation and learning are complementary behaviors, and that they can be used simultaneously (for purposes of automatic control) in a synergistic fashion. These points will be further brought out in Chapters 4 and 5.

3.5 Learning as Function Synthesis

For a wide and important class of learning control problems, the desired mapping is known (or assumed) to be continuous in advance. In such situations, memory implementations with efficient storage mechanisms can be proposed. By assuming that the desired mapping $\mathbf{M}^* : \mathbf{x} \rightarrow \mathbf{y}^*$ is continuous, an approximate mapping $\mathbf{M} : \mathbf{x} \rightarrow \mathbf{y}$ can be implemented by any scheme capable of approximating arbitrary continuous functions. In such cases, the mapping \mathbf{M} is represented as a continuous function, parameterized by a vector \mathbf{p} ; i.e., $\mathbf{M} = \mathbf{M}(\mathbf{x}; \mathbf{p})$. The learning update step would be achieved by appropriately adjusting the parameter vector \mathbf{p} by an amount $\Delta \mathbf{p}$ (yet to be determined). By "appropriate," we mean that the adjusted parameter vector $\bar{\mathbf{p}} = \mathbf{p} + \Delta \mathbf{p}$ is such that the resulting $\bar{\mathbf{y}} = \mathbf{M}(\mathbf{x}; \bar{\mathbf{p}})$ would be "better" than the original \mathbf{y} , relative to the desired response \mathbf{y}^* . As new learning experiences became available, the mapping \mathbf{M} would be incrementally im-

proved. Recall would be achieved by evaluating the functional mapping at a particular point in its input domain.

In this parameterized approach to function synthesis, the knowledge that is gained over time is stored in a *distributed* fashion in the parameter space of the mapping. This feature, which arises naturally in any practical implementation of a continuous mapping, can be most desirable from a learning control point of view (depending on the way it is achieved, as discussed below). Distributed learning is advantageous when previous learning under similar circumstances can be *combined* to provide a suitable response for the current situation. This fusion process effectively broadens the scope and influence of each learning experience and is referred to as *generalization*.

There are several important ramifications of generalization. First, it has the effect of eliminating "blank spots" in the memory (i.e., specific points at which no learning has occurred), since *some* response (albeit not necessarily the desired one) will always be generated. Second, it has the effect of constraining the set of possible input/output mappings that can be achieved by the mapping, since in most cases neighboring input situations will result in similar outputs. Finally, generalization complicates the learning process, since the adjustment of the mapping following a learning experience cannot be considered as an independent, point-by-point process (e.g., as in BOXES [Michie & Chambers (1968)]). In spite of this, the advantages accorded by generalization usually far outweigh the difficulties it evokes.

Generalization is an intrinsic feature of function synthesis approaches that rely on parameterized continuous mappings. In any practical implementation having a finite number of adjustable parameters, each adjustable parameter will affect the realized function over a region of nonzero measure. When a single parameter p_j (from the set $\mathbf{p} = \{p_1, p_2, \dots, p_N\}$) is adjusted to improve the approximation at a specific point \mathbf{x} , the continuous mapping \mathbf{M} (i.e., at least one of the outputs of $\mathbf{M} = \{M_1, M_2, \dots, M_m\}$) will be affected throughout the region of "influence" of that parameter. This region of influence is determined by the partial derivatives $\partial M_i / \partial p_j$ (one for each output of \mathbf{M}), which are functions of the input \mathbf{x} . Under these conditions, the effect of a learning experience will be generalized automatically, and extended to all parts of the mapping in which the "sensitivity"

functions $\partial M_i / \partial p_j$ are nonzero. The greatest effect will occur where $|\partial M_i / \partial p_j|$ is largest; little or no change will occur wherever this quantity is small or zero. The nature of this generalization may or may not be beneficial to the learning process depending on whether the extent of the generalization is local or global. These issues are further discussed in Subsection 3.5.3.

For function synthesis approaches based on parameterized representations, the learning process requires an algorithm that will specify an appropriate $\Delta \mathbf{p}$ so as to achieve some desired objective. When the mathematical structure used to implement the mapping is continuously differentiable and the objective function J can be treated as a "cost" to be minimized, then the construction of $\Delta \mathbf{p}$ can be straightforward. In the special case where the adjustable parameters \mathbf{p} appear linearly in the gradient vector $\partial J / \partial \mathbf{p}$ of the cost function J with respect to the adjustable parameters \mathbf{p} , the optimization could be treated as a linear algebra problem; in general (i.e., for most applications), nonlinear optimization methods *must* be used. One nonlinear technique that is suitable for on-line learning is the *gradient learning algorithm*: $\Delta \mathbf{p} = -\mathbf{W} \cdot (\partial J / \partial \mathbf{p})^T$, where \mathbf{W} is a symmetric positive definite matrix that determines the "learning rate," and the gradient $\partial J / \partial \mathbf{p}$ is defined to be a row vector. If a second-order Taylor expansion is used to provide a local approximation of the objective function J (about the current parameter vector \mathbf{p}), then the "optimum" \mathbf{W} which minimizes this local quadratic cost function in a single step can be shown to be equal to the inverse of the Hessian matrix \mathbf{H} (of J), so that

$$\mathbf{W}_{opt} = \mathbf{H}^{-1} = \left(\frac{\partial^2 J}{\partial \mathbf{p} \partial \mathbf{p}^T} \right)^{-1} \quad (3.1)$$

This equation is only valid when the local Hessian matrix is positive definite. Because it is difficult to compute and invert the Hessian on-line, the weight matrix \mathbf{W} is usually only an approximation of the full Hessian, as in the Levenberg-Marquardt method [Press, et al. (1988)]. Often, in fact, a single learning rate coefficient α is used to set $\mathbf{W} = \alpha \mathbf{I}$.

More insight can be gained into the gradient learning algorithm through an application of the chain rule, which yields: $\Delta \mathbf{p} = -\mathbf{W} \cdot (\partial \mathbf{y} / \partial \mathbf{p})^T \cdot (\partial J / \partial \mathbf{y})^T$ (where the Jacobian $\partial \mathbf{y} / \partial \mathbf{p}$ is defined as a matrix of gradient row vectors $\partial y_i / \partial \mathbf{p}$, so that $\partial J / \partial \mathbf{p} = (\partial J / \partial \mathbf{y}) \cdot (\partial \mathbf{y} / \partial \mathbf{p})$). This form of the gradient learning rule involves two

types of information: the Jacobian of the outputs of the mapping with respect to the adjustable parameters, and the gradient of the objective function with respect to the mapping outputs. The gradient $\partial J/\partial \mathbf{y}$ is determined both by the specification of the objective function J and the manner in which the mapping outputs affect this function (which, in turn, is determined by the way in which the learning system is used within the control system architecture). The Jacobian $\partial \mathbf{y}/\partial \mathbf{p}$ is completely determined by the approximation structure \mathbf{M} and, hence, is known *a priori* as a function of the input \mathbf{x} . Note that the performance feedback information provided to the learning system is the output gradient $\partial J/\partial \mathbf{y}$. This gradient vector provides the learning system with considerably more information than the scalar J ; in particular, $\partial J/\partial \mathbf{y}$ indicates both a direction and magnitude for $\Delta \mathbf{p}$ (since $\partial \mathbf{y}/\partial \mathbf{p}$ is known), whereas performance feedback based solely on the scalar J does neither.

To give an illustrative example, a simple quadratic objective function might be defined as

$$J = \frac{1}{2} \sum_{\mathbf{E}} \mathbf{e}_i^T \mathbf{e}_i \quad (3.2)$$

where J is the cost to be minimized (over a finite set of evaluation points $\mathbf{x}_i \in \mathbf{E} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_R\}$) and the output errors $\mathbf{e}_i = \mathbf{y}_i^* - \mathbf{y}_i = \mathbf{M}^*(\mathbf{x}_i) - \mathbf{M}(\mathbf{x}_i)$ are assumed to be known. In the special case where the objective function is given by (3.2) and $\mathbf{W} = \alpha \mathbf{I}$, the learning rule is

$$\Delta \mathbf{p} = \alpha \sum_{\mathbf{E}} \frac{\partial \mathbf{y}_i^T}{\partial \mathbf{p}} \cdot \mathbf{e}_i$$

If the objective function is a strictly convex function of \mathbf{p} , then the gradient algorithm will find the optimum value \mathbf{p}^* that minimizes J . For most practical learning control problems, however, the situation is much more complicated. The objective function J to be minimized may involve terms that are only known implicitly (e.g., the desired output \mathbf{y}^* may not be explicitly known or, equivalently, the output error \mathbf{e} of the mapping may not be measurable); moreover, J may be significantly more complex than that shown in (3.2) (e.g., J may be a dynamic rather than a static function). Finally, for reasons that will be discussed in Subsection 3.5.2, objective functions defined over a finite set of evaluation points (as in (3.2)) cannot usually be used directly for on-line learning control.

As with all gradient based optimization techniques, there exists a possibility of converging to a local minimum if the objective function is not convex. This point together with the preceding discussion suggests two desiderata for learning control systems employing gradient learning methods: first, the architecture should allow for the determination (or accurate estimation) of the gradient $\partial J/\partial \mathbf{y}$ and, second, the cost function J should be a convex function of the adjustable parameters \mathbf{p} . Note that it may be possible to determine or estimate $\partial J/\partial \mathbf{y}$ without ever knowing \mathbf{y}^* .

3.5.1 Connectionist Learning Systems

Connectionist systems, including what are often called "artificial neural networks," have been suggested by many authors to be ideal structures for the implementation of learning control systems. A typical connectionist system is organized in a network architecture that is comprised of nodes and connections between nodes. Each node can be thought of as a simple processing unit, with a number of adjustable parameters (which do not have to appear linearly in the nodal input/output relationship). Typically, the number of different node types in a network is small compared to the total number of nodes. Common examples include multilayer sigmoidal [Rumelhart, Hinton, & Williams (1990)] and radial basis function [Poggio & Girosi (1990)] networks.¹ The popularity of such systems arises, in part, because they are relatively simple in form, are amenable to gradient learning methods, and can be implemented in parallel computational hardware.

Perhaps more importantly, however, it is well known that several classes of connectionist systems have the *universal approximation property*. This property implies that any continuous function can be approximated to a given degree of accuracy by a sufficiently large network [Funahashi (1989); Hornik, Stinchcombe, &

¹ We do not consider any *recurrent* networks (i.e., networks having internal feedback and, hence, internal state) in this discussion for the simple reason that any recurrent network representing a continuous or discrete-time dynamic mapping can be expressed as an equivalent *dynamical system* comprised of two *static* mappings separated by either an integration or unit delay operator. In other words, the problem can always be decomposed into two component problems: that of estimating the parameters of the static mappings and that of estimating the state of the dynamical system (e.g., via an extended Kalman filter [Livstone, Farrel, & Baker (1992)]).

White (1989)]. Although the universal approximation property is important, it is held by so many different approximation structures that it does not form a suitable basis upon which to distinguish them. Thus, we must ask what other attributes are important in the context of learning control. In particular, we must look beyond the initial biological motivations for connectionist systems and determine whether they indeed hold any advantage over more traditional approximation schemes. An important factor to consider is the *environment* in which learning will occur. Thus, for example, the quantity, quality, and content of the information that is likely to be available to the learning system during its operation critically impact its performance, and should be accounted for in the selection of a suitable learning approach.

The particular scenarios that we will consider involve the use of *passive* learning strategies; that is, learning schemes that are opportunistic and exploit whatever information happens to be available during the normal course of operation of the closed-loop system. In contrast, one might also consider *active* learning strategies, in which the learning control system not only attempts to drive the outputs of the plant along a desired trajectory, but also explicitly seeks to improve the accuracy of the mapping maintained by the learning system. This is achieved by introducing "probing" signals that direct the plant into regions of its state-space where insufficient learning has occurred. Active learning control is analogous to *dual* (adaptive) control [Åström & Wittenmark (1989)]. Because we wish to focus on passive learning strategies, the learning systems we consider must be capable of accommodating on-line measurements and performance feedback that arise during the normal operation of the closed-loop system. This situation presents special challenges, as discussed in the next subsection.

3.5.2 Incremental Learning Issues

If the goal is to have learning occur on-line, in conjunction with a plant that can be nominally modeled as the discrete-time dynamical system

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k)\end{aligned}\tag{3.3}$$

where $\mathbf{f}(\cdot, \cdot)$ and $\mathbf{h}(\cdot, \cdot)$ are continuous, then an objective function of the form given by (3.2) cannot be used directly. The main problem is that the set of possible inputs

to the mapping maintained by the learning system will not consist of a finite set of discrete points. Consequently, it will not be easy way to select a finite set of representative evaluation points $\mathbf{z}_i \in \mathbf{E}$, nor will it be possible to guarantee that any or all of them are ever visited. In general, the inputs \mathbf{z} to the learning system will be comprised from measured or estimated values of $\{\mathbf{x}, \mathbf{u}, \mathbf{y}\}$ —which represent a continuum. Fortunately, various alternative objective functions that approximate (3.2) are feasible and are often used in practice. For example, one approach would be to allow the set \mathbf{E} to grow on-line to include all \mathbf{z}_i as they are encountered; i.e.,

$$\mathbf{E}_k = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\} \quad (3.4)$$

In the special case where the adjustable parameters \mathbf{p} appear linearly in the gradient $\partial J / \partial \mathbf{p}$ of (3.2) and \mathbf{E} is given by (3.4), recursive linear estimation techniques (e.g., RLS) could be used to obtain the "optimum" parameter vector \mathbf{p}^* (corresponding to the particular set \mathbf{E}). In most connectionist networks, however, some or all of the adjustable parameters appear nonlinearly in $\partial J / \partial \mathbf{p}$; hence, linear optimization methods cannot be used. Moreover, evaluation sets of the form given by (3.4) are difficult to employ in a nonlinear setting.

By far, the most common objective function used for on-line learning in control applications is the *point-wise* function given by

$$J = \frac{1}{2} \mathbf{e}^T \mathbf{e} \quad (3.5)$$

(3.5) can be considered as a special case of (3.2) when the evaluation set \mathbf{E} contains a single point at each sampling instant. Learning algorithms that seek to minimize point-wise objective functions in lieu of objective functions defined over a continuum are referred to as *incremental* learning algorithms; they are related to a broad class of *stochastic approximation* methods [Gelb (1974)]. *Incremental gradient* learning algorithms operate by approximating the actual gradient $\partial J / \partial \mathbf{p}$ of (3.2) with an *instantaneous* estimate of the gradient, based on (3.5). Incremental gradient learning algorithms of this form are related to *stochastic gradient* methods [Haykin (1991)]. The use of point-wise objective functions to approximate *batch* (or *ensemble*) objective functions (i.e., those in which \mathbf{E} contains more than one point) will generally not be successful unless special attention is given to the distribution of the evaluation points, the form of the learning algorithm, and the structure of the network. We will have more to say concerning this point in the next subsection.

One well-known and widely used stochastic gradient algorithm is the least-mean-square (LMS) algorithm [Widrow & Hoff (1960)]. The LMS parameter adjustment law is $\Delta \mathbf{p} = -\alpha(\partial J/\partial \mathbf{p})^T$, where the gradient $\partial J/\partial \mathbf{p}$ is based on (3.5). Given certain assumptions (e.g., linearity, stationarity, Gaussian-distributed random variables, etc.), LMS can be shown to be convergent, relative to the objective function of (3.2), with \mathbf{E} given by (3.4). In this case, the LMS algorithm is guaranteed to be *convergent in the mean and mean-square*, i.e.,

$$\lim_{k \rightarrow \infty} E(\mathbf{p}_k) = \mathbf{p}_{opt} \quad \text{and} \quad \lim_{k \rightarrow \infty} E(J_k) = J_{subopt} > J_{min}$$

where $E(\cdot)$ denotes expected value, if the learning rate coefficient α (a constant) satisfies conditions related to the eigenvalues of the input correlation matrix of (e.g., α cannot be too large) [Haykin (1991)]. In the first limit, as the number of learning experiences goes to infinity, the expected value of the parameter vector approaches that of the optimum parameter vector \mathbf{p}_{opt} , corresponding to the Wiener solution for this problem (which achieves J_{min}). In the second limit, the expected value of the cost (which is the mean-square error), also approaches a limit, but not the minimum value achieved by the optimum (Wiener) solution. Under these same conditions, convergence of the parameter vector (not its expected value) to the optimum value, i.e.,

$$\lim_{k \rightarrow \infty} \mathbf{p}_k = \mathbf{p}_{opt}$$

can be obtained if the learning rate coefficient decreases at a special rate over time (e.g., $\alpha_k \sim 1/k$) [Gelb (1974)]. Although the theory supporting the stability and convergence of the LMS algorithm only applies to the special case of a linear network (among other assumptions), the basic strategy underlying LMS has been used to formulate a simple learning algorithm for nonlinear networks. In this case, the parameter adjustment law becomes

$$\Delta \mathbf{p} = \alpha \frac{\partial \mathbf{y}^T}{\partial \mathbf{p}} \cdot \mathbf{e} \quad (3.6)$$

where $\partial J/\partial \mathbf{p}$ is based on (3.5) (with $\mathbf{e} = \mathbf{y}^* - \mathbf{y}$). (3.6) represents the standard incremental gradient algorithm used by many practitioners for on-line learning control.

3.5.3 Spatially Localized Learning

Special constraints are placed on a learning system whenever learning is to occur on-line, during closed-loop operation; these constraints can impact the network architecture, learning algorithm, and training process. Assuming a passive learning system is being employed, the learning experiences (training examples) cannot be selected freely, since the plant state (and outputs) are constrained by the system dynamics, and the desired plant outputs are constrained by the specifications of the control problem (without regard to learning). Under these conditions, the system state may remain in small regions of its state-space for extended periods of time (e.g., near setpoints). In turn, this implies that the data z used for incremental learning will remain in small regions of the input domain of the mapping being synthesized. Such stasis can cause undesirable side-effects in situations where parameter adjustments (based on incremental learning algorithms) have a nonlocal effect on the mapping maintained by the learning system.

For example, if a parameter that has a nonlocal effect on the mapping is repeatedly adjusted to correct the mapping in a particular region of the input domain, this may cause the mapping in other regions to deteriorate and, thus, can effectively "erase" learning that has previously taken place. Such undesirable behavior arises because the parameter adjustments dictated by an incremental learning algorithm are made on the basis of a single evaluation point, *without regard to the remainder of the mapping*. Another unfortunate phenomenon is inherent in all incremental learning algorithms: *conflicting demands on the adjustable parameters* are created because, for instance, the vector \mathbf{p}_i^* that minimizes J in (3.5) at some point \mathbf{z}_i , will generally differ from the vector \mathbf{p}_j^* that minimizes this function at some other point \mathbf{z}_j . The idiosyncrasies associated with passive incremental learning in closed-loop control (i.e., stasis coupled with nonlocal learning, and conflicting parameter updates), have precipitated the development and analysis of *spatially localized* learning systems.

The basic idea underlying spatially localized learning arises from the observation that learning is facilitated in situations where a clear association can be made between a subset of the adjustable elements of the learning system and a localized region of the input-space. Further consideration of this point in the context of the difficulties described above, suggests several *desired* traits for learning systems

that rely on incremental gradient learning algorithms. These objectives can be expressed in terms of the previously mentioned "sensitivity" functions $\partial M_i / \partial p_j$, which are the partial derivatives of the mapping outputs M_i with respect to the adjustable parameters p_j . At each point \mathbf{x} in the input domain of the mapping, it is desired that the following properties hold:

- for each M_i , there exists at least one p_j such that the function $|\partial M_i / \partial p_j|$ is relatively large in the vicinity of \mathbf{x} (*coverage*)
- for all M_i and p_j , if the function $|\partial M_i / \partial p_j|$ is relatively large in the vicinity of \mathbf{x} , then it should be relatively small elsewhere (*localization*)

Under these conditions, incremental gradient learning is supported throughout the input domain of the mapping, but its effects are limited to the local region in the vicinity of each learning point. Thus, experience and consequent learning in one part of the input domain have only a marginal effect on the knowledge that has already been accrued in other parts of the mapping. For similar reasons, problems due to conflicting demands on the adjustable parameters are also reduced.

Several existing learning system designs, including BOXES [Michie & Chambers (1968)], CMAC [Albus (1975)], radial basis function networks [Poggio & Girosi (1990)], and local basis/influence function networks [Baker & Farrell (1990); Millington (1991)], generally do exhibit the spatially localized learning property. In contrast, the ubiquitous sigmoidal (or perceptron) network often does not exhibit this property. To combat the problems associated with nonlocalized learning and conflicting parameter updates, a number of simple corrective procedures have been used with sigmoidal networks, including local batch learning, very slow learning rates, distributed (uncorrelated) input sequences, and randomizing input buffers (e.g., see [Baird & Baker (1990)]).

To give a simple example of spatially localized learning, we will briefly describe local basis/influence function networks and, in particular, the *linear-Gaussian network*. This approach relies on a combination of *local basis* and *influence* function nodal units to achieve a compromise between the spatially localized learning properties of quantized learning systems (e.g., those based on "bins") and the efficient representation and generalization capabilities of other connectionist networks. The complete network mapping is constructed from a set of local basis functions $f_i(\mathbf{x})$ that have applicability only over spatially localized regions of the

input-space. The influence functions $\gamma_i(\mathbf{x})$ are coupled in a one-to-one fashion with the basis functions $\mathbf{f}_i(\mathbf{x})$, and are used to describe the domain over the input-space (the "sphere of influence") of each local basis function. In other words, relative to some point \mathbf{x}^0 in the input domain, each influence function $\gamma_i(\mathbf{x})$ is defined as a nonnegative function, with a maximum at \mathbf{x}^0 , that tends to zero for all points \mathbf{x} that are "far away" from \mathbf{x}^0 . The overall input/output relationship is given by

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^n \Gamma_i(\mathbf{x}) \mathbf{f}_i(\mathbf{x}) \quad (3.7a)$$

where n is the number of nodes in the network and $\Gamma_i(\mathbf{x})$ are the *normalized* influence functions, defined to be

$$\Gamma_i(\mathbf{x}) = \frac{\gamma_i(\mathbf{x})}{\sum_{j=1}^n \gamma_j(\mathbf{x})} \quad \text{with} \quad 0 < \Gamma_i(\mathbf{x}) \leq 1 \quad \text{and} \quad \sum_{i=1}^n \Gamma_i(\mathbf{x}) = 1 \quad (3.7b)$$

By design, each adjustable parameter in this network affects the overall mapping only over the limited region of its input-space described by the associated (normalized) influence function. Thus, the aforementioned stasis problem is minimized. Note also that (local) generalization is an inherent property of the network, and that standard incremental gradient learning methods can still be used.

To further illustrate the basic concept, we will consider a specific realization employing linear functions (with an offset, so that they are really affine) as the local basis units, and Gaussian functions as the influence function units. In this *linear-Gaussian* network, the functions $\mathbf{f}_i(\mathbf{x})$ and $\gamma_i(\mathbf{x})$ are defined to be:

$$\begin{aligned} \mathbf{f}_i(\mathbf{x}) &= \mathbf{M}_i(\mathbf{x} - \mathbf{x}_i^0) + \mathbf{b}_i \\ \gamma_i(\mathbf{x}) &= c_i \exp\left\{-\left(\mathbf{x} - \mathbf{x}_i^0\right)^T \mathbf{Q}_i(\mathbf{x} - \mathbf{x}_i^0)\right\} \end{aligned} \quad (3.8)$$

where, for each node pair i in the network, the matrices \mathbf{M}_i and \mathbf{Q}_i , the vectors \mathbf{x}_i^0 and \mathbf{b}_i , and the scalar c_i are all potentially adjustable (\mathbf{Q}_i must be symmetric positive definite). The vector \mathbf{x}_i^0 represents the local origin shared by the linear-Gaussian pair, the idea being that the overall mapping is approximated by $\mathbf{f}_i(\mathbf{x})$ in the "vicinity" of \mathbf{x}_i^0 (as characterized by $\Gamma_i(\mathbf{x})$, relative to all other $\Gamma_{j \neq i}(\mathbf{x})$). Because of its unique structure, physical meaning is more easily attributed to each parameter and to the overall structure of the network. As a result, *a priori* knowledge and partial solutions are easily incorporated (e.g., linear control point

designs corresponding to the $f_i(\mathbf{x})$). In fact, linear functions were chosen as the local basis units due to their simplicity and compatibility with conventional gain scheduled mappings (alternative local basis units may be more desirable if certain *a priori* knowledge is available about the regional functional structure of the desired mapping). Due to its special structure, this network also allows on-line variable structure learning schemes to be used, where nodal unit pairs can be added or removed from the network to achieve more accurate or more efficient mappings. An example of a simple linear-Gaussian network comprised of 5 pairs of local basis/influence function units is shown in Fig. 3.3; the influence functions (lower part of the figure) have been separated from each other somewhat so that each of the local linear functions is clearly visible in the overall input/output mapping (upper part of the figure).

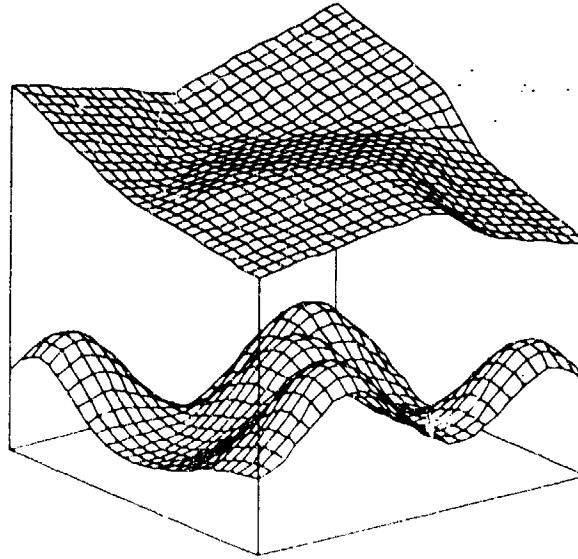


Figure 3.3. An Example of a Linear-Gaussian Network Mapping ($\mathcal{R}^2 \rightarrow \mathcal{R}$), Together with Its Underlying Influence Functions.

Learning algorithms for spatially localized networks can capitalize on localization in two ways. First, spatial localization implies that at each instant of time only a small subset of the nodal units (and hence a small subset of the adjustable parameters) have a significant effect on the network mapping. Thus, the efficiency of both calculating the network outputs and of updating the network parameters can be improved by ignoring all "insignificant" nodal units. For example, this can be realized in a linear-Gaussian network by utilizing only those

nodal unit pairs with the largest normalized influences; that is, those whose combined (normalized) influence equals or exceeds some predefined threshold (e.g., 0.95). This approach can greatly increase the throughput of a network when implemented in sequential computational hardware. Furthermore, since the system state may remain in particular regions of its state-space for extended periods of time, it is expected that the approximation error will not tend *uniformly* to zero. Instead, the error will be lowest in those areas where the greatest amount of learning has occurred. This leads to conflicting constraints on the learning rate: it should be small, to filter the effects of noise, in those regions where the approximation error is small; at the same time, it should be larger, for fast learning, in those regions where the approximation error is large (relative to the ambient noise level). Resolution of this conflict is possible through the use of *spatially localized learning rates*, where individual learning rate coefficients are maintained for each (spatially localized) adjustable parameter and updated in response to the local learning conditions. In this case, the elements of the weighting matrix W would vary individually over time.

The computational memory requirements for spatially localized networks fall somewhere between those for nonlocal connectionist networks (on the low side) and those for discrete-input, analog-output mapping architectures (on the high side). By requiring each parameter to have only a localized effect on the overall mapping, we should expect an increase in the number of parameters required to obtain a mapping comparable in accuracy to a (potentially more efficient) non-local technique. Nevertheless, for automatic control applications, training speed and approximation accuracy should have priority over memory requirements, since memory is generally inexpensive relative to the cost of inaccurate or inappropriate control actions.

3.6 Application Issues

One strategy for using learning in flight control involves the development of an off-line flight control system design that is optimized (via adaptive and learning augmentation) relative to a *design* model, followed by *in-flight* evaluation and subsequent on-line tuning relative to the *actual* vehicle. This requires that a number of key design issues be resolved. Such issues are identified and briefly

discussed below, under the assumption that a hybrid adaptive/learning control approach will be used.

Performance Requirements

As part of any control design, the desired transient and steady-state dynamical characteristics of the vehicle must be specified. For a complex system with significant nonlinearity and modeling uncertainty, the specification of performance requirements that are achievable throughout the flight envelope may be a non-trivial matter. It will almost certainly be the case that the vehicle is fundamentally capable of "delivering more" in some flight conditions than in others. Unfortunately, determining what these innate advantages are and how they can be exploited may not be completely evident until after in-flight testing with the actual vehicle has begun. On-line learning could be used to optimize the closed-loop performance of the vehicle over the entire envelope.

A Priori Control Law Design

A basic control law for the flight control system must be selected that is expected to be able to achieve the desired performance requirements. This involves the selection of the measurements to be used for feedback, the type of control law structure to be used (e.g., a simple linear combination of the feedback variables or dynamic compensation), and a nominal set of control law parameters (e.g., feedback gains). If a nominal gain scheduled controller already exists, then this design could be incorporated as *a priori* knowledge in a suitable learning system [Baker & Farrell (1992)]. Importantly, the *a priori* control law structure should be flexible enough so that both adaptive and learning augmentation can be used. Finally, an estimator/observer may be needed to synthesize and/or filter the state estimates.

Adaptive System

An adaptive control system must be selected that is an extension of the *a priori* control law design; that is, the adaptive design should be such that, if there were no modeling error, the adaptive system would contribute nothing. Either a direct or indirect adaptive control system might be used. The ability of the hybrid system

to be robust to transients caused by uncertainty is determined primarily by the *a priori* and adaptive components of the control system.

Posterior Estimator

Depending on whether a direct or indirect adaptive control system has been selected, a current estimator will exist for the adjustable control or model parameters, respectively. Since the learning process itself need not be performed in phase with, or even at the same pace as, the control update cycle, and since significantly improved estimates can be achieved by using delayed estimation methods (e.g., combinations of both filtering and smoothing), a *posterior* estimator must be developed and used to provide relatively high quality information to the learning system.

Learning System

The detailed structure and parameter update algorithm(s) of the learning system must be selected. These choices are mediated by the anticipated characteristics (e.g., number of inputs, number of outputs, and complexity) of the functional mapping(s) to be learned. If, for instance, a conventional indirect adaptive controller is used that explicitly estimates model parameters on-line, and if it can be safely assumed that the significant spatial dependencies in the dynamical behavior of the vehicle are functions of a subset of the vehicle state (e.g., altitude, speed, and angle-of-attack), then a mapping from these scheduling variables to the model parameters is required. Any *a priori* knowledge of the functional relationships between the inputs and outputs of the desired mapping could be exploited by selecting an appropriate initial structure and parameter set.

Training Procedure

The development of an appropriate (off-line) training procedure is an important subtask. The trajectories and initial conditions used for training purposes must provide sufficient excitation of the vehicle dynamical characteristics and must adequately explore the specified vehicle operating envelope. An additional issue concerns the use of a stochastic vehicle model (to approximate a range of dynamical behaviors) during training. It is possible that such stochastic behavior will

cause the off-line learning process to be less sensitive to slight modeling errors, at the expense, however, of total training time and initial closed-loop system performance.

Evaluation Procedure

In normal practice, a *design model* is used to support the flight control system design, development, and preflight evaluation, while the *actual vehicle* is used to evaluate the in-flight performance of the closed-loop system. Thus, an *off-line* evaluation of the robust capability of any approach can only be made relative to a simulation that accounts for the expected differences (due to uncertainty) between the "design" model and the "actual" vehicle. To fully demonstrate the potential benefits of a hybrid approach, in particular the on-line tuning and performance enhancement capabilities, an off-line evaluation procedure would have to allow sufficient time for "in-flight" interaction with the actual vehicle.

3.7 Expected Benefits

Potential benefits that may be accorded by learning augmentation are described below.

Control System Design and Tuning

Learning augmentation may facilitate the design and tuning of flight control systems for high performance aircraft in several ways. Learning systems can provide design automation; they can allow known nonlinearities and spatial dependencies to be compensated for directly and, in addition, they can provide a means for off-line flight control system design optimization. These benefits may result in less manual involvement during the initial design phase (to achieve a specified level of performance); in addition, a smaller number of test flights (and associated tuning) may be required for similar reasons. The net effect is a reduction in effort and, consequently, a reduction in cost. Learning could also play a corresponding role in the retrofitting of advanced flight control systems into existing high performance aircraft.

On-Line Accommodation of Uncertainty

Learning augmentation can provide an *on-line* approach for accommodating both parametric and structural model uncertainty. This is in contrast to the standard *off-line* design approach using fixed-parameter, robust control designs. With on-line learning, the level of uncertainty may be reduced through direct, closed-loop interactions with the vehicle and its environment to achieve *a posteriori* levels of uncertainty that are substantially lower than *a priori* ones. With the off-line approach, closed-loop system performance is generally sacrificed to ensure that the vehicle satisfies some minimum level of performance requirements for all likely variations of the model parameters. The tradeoff between performance and robustness may be especially severe for vehicles that are required to operate with a variety of payloads or configurations. The ability to learn on-line can reduce the *a priori* robustness required, allowing the designer to obtain a higher level of overall closed-loop system performance. Depending on the type of learning system used, it may be possible to initialize the system with an *a priori* robust control design, and allow on-line learning to improve this nominal design as the level of uncertainty is reduced through direct interaction with the vehicle.

Closed-Loop System Performance

Learning augmentation can provide an automatic mechanism for improving the level of closed-loop system performance that is ultimately achieved, through on-line self-optimization. This, together with the fact that learning systems are capable of realizing general multivariable functional mappings, means that initially untapped vehicle superiorities (e.g., agility) might be exploited to provide enhanced maneuverability.

Operational Efficiency

Since on-line learning can be used to accommodate initially unknown spatial dependencies, some transient effects that would otherwise be associated with parameter adjustment in an entirely adaptive system can be minimized to improve the operational efficiency and performance of a hybrid control system. On-line learning can reduce the burden on the adaptive system of continuously reacting to predictable nonlinearities.

4 Hybrid Adaptive/Learning Control

A specific indirect hybrid adaptive/learning control methodology is derived mathematically in this chapter. The motivation for this derivation comes from our earlier analysis of different learning control system architectures and from a consideration of the benefits of exploiting *a priori* design knowledge and of utilizing on-line adaptation. The discussion will cover three different controllers that are each based on model-reference compensators of increasing sophistication: a simple linear compensator, an adaptive version of the linear compensator, and a hybrid adaptive/learning version of the linear compensator. Thus, the overall learning augmented control system will be derived by enhancing a simple linear compensator with both adaptive and learning capabilities.

The model-based *linear* compensator was designed following a procedure similar to that in [Anderson & Schmidt (1991)]. We further developed this basic approach so that it might be applied to nonlinear problems. Subsequently, an *adaptive* compensator was developed by incorporating and extending ideas presented in [Youcef-Toumi & Ito (1990)]. Finally, a *hybrid* adaptive/learning control system was developed by combining the same adaptive compensator with a learning system, and treating the entire problem in a nonlinear framework. The detailed mathematical derivation is presented in Section 4.1, while several applications of this control methodology are summarized in Section 4.2.

4.1 Controller Development

In the following derivation, we will assume that the plant dynamics are given in discrete-time by

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k)\end{aligned}\tag{4.1}$$

where $\mathbf{x}, \mathbf{u}, \mathbf{y}$ represent the states, inputs, and outputs, respectively, of the plant, and $\dim(\mathbf{x}) = n$, $\dim(\mathbf{u}) = m$, and $\dim(\mathbf{y}) = m$. For the purpose of deriving the baseline linear compensator, we will assume that these equations have been linearized about an equilibrium point to yield a local approximation of the form

$$\begin{aligned}\mathbf{x}_{k+1} &= \Phi \mathbf{x}_k + \Gamma \mathbf{u}_k \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k\end{aligned}\quad (4.2)$$

In fact, we will limit the amount of *a priori* knowledge of the actual plant dynamics (4.1) that is available for the design of all three controllers (linear, adaptive, and hybrid) to a single, constant parameter, linear model of the form given in (4.2). Note that this is not a requirement of our approach—a nonlinear *a priori* model of the form given by (4.1) could be used without difficulty. In the derivations that follow, we chose to use linear systems to describe the reference model and desired output tracking error dynamics, although once again there is no real requirement to do so.

The development of the three compensators begins with the specification of two more dynamical systems: (i) a model of the desired closed-loop dynamics of the plant (the *reference model*) and (ii) a model of the desired dynamics of the difference between the outputs of the reference model and those of the actual plant (the *tracking error dynamics*). The output tracking error \mathbf{e} is defined as $\mathbf{e} = \mathbf{y}_r - \mathbf{y}$, where \mathbf{y}_r are the outputs of the reference model, \mathbf{y} are the outputs of the plant, and $\dim(\mathbf{e}) = m$ and $\dim(\mathbf{y}_r) = m$. The desired tracking error dynamics are defined as

$$\mathbf{e}_{k+1} = \Phi_e \mathbf{e}_k \quad (4.3)$$

where Φ_e is a matrix with all eigenvalues inside the unit circle (i.e., so that (4.3) is a stable system). Other forms of the error dynamics (4.3) can also be pursued (e.g., by the addition of integral action). Finally, we will assume that the desired reference model is also linear and stable:

$$\begin{aligned}\mathbf{x}_{r,k+1} &= \Phi_r \mathbf{x}_{r,k} + \Gamma_r \mathbf{r}_k \\ \mathbf{y}_{r,k} &= \mathbf{C}_r \mathbf{x}_{r,k}\end{aligned}\quad (4.4)$$

where \mathbf{x}_r and \mathbf{r} represent the states and inputs, respectively, of the reference model, and $\dim(\mathbf{x}_r) = p$ and $\dim(\mathbf{r}) = m$.

4.1.1 Baseline Linear Compensator

The linearized plant model can be expressed in an input-output form by collapsing the state-space description of (4.2):

$$\mathbf{y}_{k+1} = \mathbf{C} \Phi \mathbf{x}_k + \mathbf{C} \Gamma \mathbf{u}_k \quad (4.5)$$

A similar operation can be performed to produce an input-output description of the reference model:

$$\mathbf{y}_{r,k+1} = \mathbf{C}_r \Phi_r \mathbf{x}_{r,k} + \mathbf{C}_r \Gamma_r \mathbf{r}_k \quad (4.6)$$

Now, given (4.3), (4.5), and (4.6), it is possible to summarize the control objective: assuming (4.2) is minimum phase (the possibility that it is not is discussed below), we want to pick \mathbf{u}_k so that (if possible) the following equation holds

$$\Phi_e \mathbf{e}_k = \mathbf{y}_{r,k+1} - \mathbf{y}_{k+1} \quad (4.7)$$

where the quantity $\mathbf{y}_{des,k+1} \equiv \mathbf{y}_{r,k+1} - \Phi_e \mathbf{e}_k$ becomes the overall desired response of the aircraft at time $k+1$ to the applied controls \mathbf{u}_k . Thus, at each time k we seek a \mathbf{u}_k that satisfies

$$\Phi_e \mathbf{e}_k = \mathbf{C}_r \Phi_r \mathbf{x}_{r,k} + \mathbf{C}_r \Gamma_r \mathbf{r}_k - \mathbf{C} \Phi \mathbf{x}_k - \mathbf{C} \Gamma \mathbf{u}_k \quad (4.8)$$

Without much difficulty, a control law can be derived in a manner similar to that outlined in [Anderson & Schmidt (1991)] to achieve the objective specified by (4.7). If the matrix $(\mathbf{C}\Gamma)$ is invertible, then a potential compensator that would result in *perfect* output tracking is given by (4.4) and

$$\mathbf{u}_k = (\mathbf{C}\Gamma)^{-1} [\mathbf{y}_{r,k+1} - \mathbf{C} \Phi \mathbf{x}_k - \Phi_e \mathbf{e}_k] \quad (4.9)$$

Note that (4.4) and (4.9) together represent a linear, multivariable, model-based compensator, requiring full-state measurements (or the use of a state observer).

It must be stressed, however, that even if $(\mathbf{C}\Gamma)$ is nonsingular, *this compensator cannot be used if the zero dynamics of the open-loop system are unstable*, as this would be tantamount to attempting to cancel a nonminimum phase zero [Slotine & Li (1991)]. In many applications, it is known from the basic physics of the problem that the plant does not have any unstable transmission zeros, and so this is not an issue. In general, though, this problem can be addressed by choosing alternative tracking outputs or by utilizing a more sophisticated on-line control design technique (see below).

Noninput/Output Square Systems

If $(\mathbf{C}\Gamma)$ is singular (e.g., because the system (4.5) is not input-output square), then the previous compensator (4.9) can be modified by using the following relation

$$\mathbf{u}_k = (\mathbf{C}\Gamma)^+ [\mathbf{y}_{r,k+1} - \mathbf{C} \Phi \mathbf{x}_k - \Phi_e \mathbf{e}_k] \quad (4.10)$$

where $(\cdot)^+$ denotes the pseudo-inverse of the argument and $\mathbf{M}^+ \equiv (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T$. In this case, perfect tracking is not generally possible; nevertheless, it can be shown that (4.10) is the "best" choice for \mathbf{u}_k in the sense that it minimizes the Euclidean norm

$$\|\mathbf{y}_{des,k+1} - \mathbf{y}_{k+1}\| \quad (4.11)$$

Similar developments exist for cases where there are more controls than outputs, or where tracking error is to be balanced against control usage (see below).

Infinite-Horizon LQ Formulation

As an alternative to the direct design approach outlined above, one could instead setup and solve the following linear-quadratic regulator (LQR) design problem (where J represents the cost to be minimized) [Anderson & Schmidt (1991)]:

$$J = \frac{1}{2} \sum_0^{\infty} \left[(\mathbf{e}_{k+1} - \Phi \mathbf{e}_k)^T \mathbf{Q} (\mathbf{e}_{k+1} - \Phi \mathbf{e}_k) + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k \right]$$

subject to the constraint equations

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_{r,k+1} \\ \mathbf{r}_{k+1} \end{bmatrix} = \begin{bmatrix} \Phi & 0 & 0 \\ 0 & \Phi_r & 0 \\ 0 & 0 & \Phi_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{r,k} \\ \mathbf{r}_k \end{bmatrix} + \begin{bmatrix} \Gamma \\ 0 \\ 0 \end{bmatrix} \mathbf{u}_k$$

$$\begin{bmatrix} \mathbf{e}_k \\ \mathbf{e}_{r,k} \end{bmatrix} = \begin{bmatrix} -\mathbf{C} & \mathbf{C}_r & 0 \\ -\mathbf{C}\Phi & \mathbf{C}_r\Phi_r & \mathbf{C}_0 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{r,k} \\ \mathbf{r}_k \end{bmatrix} + \begin{bmatrix} \mathbf{v} \\ -\mathbf{C}\Gamma \end{bmatrix} \mathbf{u}_k$$

where Φ_0 represents a fictitious reference command shaping filter that does not appear in the final control law and \mathbf{Q} and \mathbf{R} are assumed to be symmetric positive definite matrices. The cost function J is motivated from the desire to simultaneously achieve the tracking error dynamics (4.3) and at the same time penalize the applied controls. The solution to this optimization problem yields the necessary gains corresponding to the state feedback from the plant as well as the feedforward terms from the model reference system and exogenous input. Although this approach involves greater computation, the solution is *guaranteed* to be stable in the linear case under very mild assumptions [Bilzon & Ho (1975); Maciejowski (1989); Stevens & Lewis (1992)]. Since minimum phase characteristics were

not generally an issue for us, we chose to use the substantially simpler on-line design method given by (4.9).

One-Step LQ Formulation

Note also that a one-step optimization involving both output and control weighting is possible. For example, if the cost to be minimized over the next time-step is

$$J_{k+1} = \frac{1}{2}(\mathbf{e}_{k+1} - \Phi_e \mathbf{e}_k)^T \mathbf{Q}(\mathbf{e}_{k+1} - \Phi_e \mathbf{e}_k) + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k$$

then the optimal solution is given by (note the similarity to (4.9) and (4.10))

$$\mathbf{u}_k = [\mathbf{R} + (\mathbf{C}\Gamma)^T \mathbf{Q}(\mathbf{C}\Gamma)]^{-1} (\mathbf{C}\Gamma)^T \mathbf{Q}[\mathbf{y}_{r,k+1} - \mathbf{C}\Phi \mathbf{x}_k - \Phi_e \mathbf{e}_k]$$

Related developments are presented on pp. 91-101, of Attachment 3.

4.1.2 Baseline Adaptive Compensator

To address the fact that there may be modeling error (i.e., that the *a priori* design model given by (4.2) may be quite different from the actual plant dynamics given by (4.1)), adaptation can be incorporated into the operation of the linear compensator. A simple technique for achieving such adaptation is described in [Youcef-Toumi & Ito (1990)]. The baseline linear compensator described above can be modified to include this capability as follows. First, we assume the *design* model for the plant is

$$\mathbf{y}_{k+1} = \mathbf{C}\Phi \mathbf{x}_k + \mathbf{C}\Gamma \mathbf{u}_k + \Psi_k \quad (4.12)$$

where Ψ_k represents the unmodeled behavior at time k . Following [Youcef-Toumi & Ito (1990)], a simple adaptive estimate for the unmodeled nonlinearities Ψ_k can be generated by solving (4.12) at the *previous* time index for Ψ_{k-1} , and then by assuming that $\Psi_k \approx \Psi_{k-1}$; this yields

$$\hat{\Psi}_k = \mathbf{y}_k - \mathbf{C}\Phi \mathbf{x}_{k-1} - \mathbf{C}\Gamma \mathbf{u}_{k-1} \quad (4.13)$$

Given this estimate for the unmodeled behavior, an adaptive compensator can be constructed in exactly the same way as (4.9) was derived. Simple algebraic manipulation yields

$$\mathbf{u}_k = (\mathbf{C}\Gamma)^{-1} [\mathbf{y}_{r,k+1} - \mathbf{C}\Phi \mathbf{x}_k - \hat{\Psi}_k - \Phi_e \mathbf{e}_k] \quad (4.14)$$

Note that the reference model update equations (4.4) do not change in this derivation, and hence are not repeated. The adaptive analog of (4.10) that minimizes the norm (4.11) can be similarly derived.

Clearly, the adaptive mechanism described by (4.13) is very crude and cannot be used to address changes in control effectiveness ($\partial \mathbf{g} / \partial \mathbf{u}$) in the actual plant, as a function of either \mathbf{x} or \mathbf{u} (here, the function $\mathbf{g}(\cdot, \cdot)$ is defined as the composition of the functions $\mathbf{h}(\cdot)$ and $\mathbf{f}(\cdot, \cdot)$ from (1); i.e. $\mathbf{g} \equiv \mathbf{h} \circ \mathbf{f}$). Moreover, the time-delay estimate (4.13) is susceptible to the presence of noise (or similar rapidly varying influences). Despite these drawbacks, which can be somewhat ameliorated by utilizing a small time step (faster sampling) and a low-pass filter of the estimate (4.13), simple adaptive compensators of this form have performed remarkably well in complex nonlinear simulations (e.g., [Millington & Baker (1992)]) involving process disturbances, sensor noise, and unmodeled aero-, engine, and actuator dynamics.

4.1.3 Hybrid Adaptive/Learning Compensator

Assuming no additional unmodeled states, the *a priori* modeling error Ψ_k will generally be a function of both the current states \mathbf{x}_k and applied controls \mathbf{u}_k . In the simple time-delay estimate (4.13) used by the adaptive compensator, the unmodeled effect of the current control vector \mathbf{u}_k on the subsequent output vector \mathbf{y}_{k+1} is not addressed in the determination of \mathbf{u}_k . This implies that the adaptive compensator is not well equipped to handle significant modeling error involving the effectiveness ($\partial \mathbf{g} / \partial \mathbf{u}$) of the control variables. In flight applications, control effectiveness can change dramatically as a function (primarily) of dynamic pressure and vehicle attitude relative to the incident wind.

To accommodate such nonlinear effects, as well as other errors due to the time-delay approximation, a learning system can be used to *synthesize* (on-line) the functional mapping described by the unmodeled nonlinearities, in terms of the key plant states, environmental variables, and applied controls. The use of learning systems in this setting is further motivated and described in [Baker & Farrell (1991); Baker & Farrell (1992); Millington & Baker (1992)].

The hybrid compensator can be derived from the baseline adaptive compensator by assuming that the *design* model for the plant has the form

$$\mathbf{y}_{k+1} = \mathbf{C}\Phi\mathbf{x}_k + \mathbf{C}\Gamma\mathbf{u}_k + \mathbf{n}(\mathbf{x}_k, \mathbf{u}_k) + \Psi_k \quad (4.15)$$

where $\mathbf{n}(\mathbf{x}_k, \mathbf{u}_k)$ represents initially unmodeled nonlinear behavior that will be "learned" by a network approximation, as a function of \mathbf{x}_k and \mathbf{u}_k . The vector Ψ_k represents any residual nonlinear behavior not captured by the *a priori* model (4.2) or through learning augmentation. The network mapping is implicitly dependent on time for the simple reason that it is evolving as a result of learning action. As before, a simple adaptive estimate for the unmodeled nonlinearities Ψ_k can be generated by solving (4.15) at the previous time index for Ψ_{k-1} , and assuming that $\Psi_k \approx \Psi_{k-1}$; this yields

$$\hat{\Psi}_k = \mathbf{y}_k - \mathbf{C}\Phi\mathbf{x}_{k-1} - \mathbf{C}\Gamma\mathbf{u}_{k-1} - \mathbf{n}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \quad (4.16)$$

Unlike the previous cases, the control objective (4.7) cannot be solved directly because it is nonlinear in \mathbf{u}_k , due to the presence of the term $\mathbf{n}(\mathbf{x}_k, \mathbf{u}_k)$. One solution to this nonlinear programming problem is to use an iterative Newton-Raphson technique [Press, *et al.* (1988)] to find \mathbf{u}_k . Linearizing (4.15) about the point \mathbf{u}_{k-1} yields

$$\mathbf{y}_{k+1} \approx \mathbf{C}\Phi\mathbf{x}_k + \mathbf{C}\Gamma\mathbf{u}_k + \mathbf{n}(\mathbf{x}_k, \mathbf{u}_{k-1}) + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \cdot (\mathbf{u}_k - \mathbf{u}_{k-1}) + \Psi_k \quad (4.17)$$

where $(\partial \mathbf{n} / \partial \mathbf{u})$ is the Jacobian matrix of $\mathbf{n}(\cdot, \cdot)$ with respect to the argument \mathbf{u} , evaluated at $\{\mathbf{x}_k, \mathbf{u}_{k-1}\}$. Using this approximation (in which \mathbf{u}_k appears linearly), it is possible to compute the first Newton-Raphson estimate \mathbf{u}_k^1

$$\mathbf{u}_k^1 = \left(\mathbf{C}\Gamma + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \right)^{-1} \left[\mathbf{y}_{r,k+1} - \mathbf{C}\Phi\mathbf{x}_k - \mathbf{n}(\mathbf{x}_k, \mathbf{u}_{k-1}) + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \cdot \mathbf{u}_{k-1} - \hat{\Psi}_k - \Phi_k \mathbf{e}_k \right] \quad (4.18)$$

assuming that the indicated matrix inversion is possible; if the matrix is singular, then a pseudo-inverse can be used instead. Subsequent estimates \mathbf{u}_k^i are obtained by relinearizing (4.15) about the points \mathbf{u}_k^{i-1} . In our work, we have found that the estimate obtained after the first iteration is sufficiently accurate, given that: \mathbf{u}_{k-1} is used as the initial guess, the discrete time step is small, and that the admissible change in the controls $\Delta \mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$ is fundamentally limited by actuator rate limits. For these reasons, we do not expect $\|\Delta \mathbf{u}_k\|$ to be very large and, thus, believe that (4.18) offers a reasonably good estimate for \mathbf{u}_k . As with the adaptive compensator, the reference model update equations do not change in this

derivation, and are not repeated. The hybrid analog of (4.10), which minimizes the norm (4.11), can also be easily derived.

Additional Remarks

It is interesting to compare the controllers that have been obtained from the baseline linear compensator, through successive augmentation by of adaptation and learning. All three perform the operations associated with updating the reference model and computing the tracking error and desired tracking output—see Table 4.1. Moreover, they all have access to the same *a priori* design model; namely, that given by (4.5). They differ, however, in the complexity of the process model considered during the on-line design procedure; in turn, this impacts the selection of the control vector to be applied. Only the hybrid scheme is capable of accommodating changes in the control effectiveness, as a function of the state and applied controls. The Jacobian matrix $(\partial \mathbf{n} / \partial \mathbf{u})$ that is added to the constant matrix (CT) accounts for such changes.

Note that in the case of the hybrid compensator, a solution that perfectly satisfies the control objective (4.7) is not guaranteed to exist even if the modified control effectiveness matrix in (4.18) is invertible. Thus, the nonlinear programming problem associated with the control selection may not have a solution; whereas in the cases of the linear or adaptive compensators, the control selection problems are linear and, hence, are guaranteed to have unique solutions under the assumed invertibility. In practice, this might occur due to physical constraints such as actuator position or rate limits that prevent the desired tracking output $\mathbf{y}_{des,k+1}$ from being achieved by any admissible control. In such cases, the desired output is said to be *unreachable* given the admissible control values.

Table 4.1. Summary of Three Related Model-Reference Compensators.

| | |
|-------------------------|---|
| reference model update | $\mathbf{x}_{r,k+1} = \Phi_r \mathbf{x}_{r,k} + \Gamma_r \mathbf{r}_k$ $\mathbf{y}_{r,k+1} = \mathbf{C}_r \mathbf{x}_{r,k+1}$ |
| tracking error dynamics | $\mathbf{e}_k = \mathbf{y}_{r,k} - \mathbf{y}_k$ $\mathbf{y}_{des,k+1} = \mathbf{y}_{r,k+1} - \Phi_e \mathbf{e}_k$ |
| process model | <p>linear $\mathbf{y}_{k+1} = \mathbf{C}\Phi\mathbf{x}_k + \mathbf{C}\Gamma\mathbf{u}_k$</p> <p>adaptive $\mathbf{y}_{k+1} = \mathbf{C}\Phi\mathbf{x}_k + \mathbf{C}\Gamma\mathbf{u}_k + \Psi_k$</p> <p>hybrid $\mathbf{y}_{k+1} = \mathbf{C}\Phi\mathbf{x}_k + \mathbf{C}\Gamma\mathbf{u}_k + \mathbf{n}(\mathbf{x}_k, \mathbf{u}_k) + \Psi_k$</p> |
| adaptive estimate | <p>adaptive $\hat{\Psi}_k = \mathbf{y}_k - \mathbf{C}\Phi\mathbf{x}_{k-1} - \mathbf{C}\Gamma\mathbf{u}_{k-1}$</p> <p>hybrid $\hat{\Psi}_k = \mathbf{y}_k - \mathbf{C}\Phi\mathbf{x}_{k-1} - \mathbf{C}\Gamma\mathbf{u}_{k-1} - \mathbf{n}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$</p> |
| learning approximation | $\mathbf{n}(\mathbf{x}_k, \mathbf{u}_k) \approx \mathbf{n}(\mathbf{x}_k, \mathbf{u}_{k-1}) + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \cdot (\mathbf{u}_k - \mathbf{u}_{k-1})$ |
| control selection | <p>linear $\mathbf{u}_k = (\mathbf{C}\Gamma)^{-1} [\mathbf{y}_{des,k+1} - \mathbf{C}\Phi\mathbf{x}_k]$</p> <p>adaptive $\mathbf{u}_k = (\mathbf{C}\Gamma)^{-1} [\mathbf{y}_{des,k+1} - \mathbf{C}\Phi\mathbf{x}_k - \hat{\Psi}_k]$</p> <p>hybrid $\mathbf{u}_k = \left(\mathbf{C}\Gamma + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \right)^{-1} \left[\mathbf{y}_{des,k+1} - \mathbf{C}\Phi\mathbf{x}_k - \mathbf{n}(\mathbf{x}_k, \mathbf{u}_{k-1}) + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \cdot \mathbf{u}_{k-1} - \hat{\Psi}_k \right]$</p> |

4.1.4 Learning System Update

The network approximation model $\mathbf{n}(\mathbf{x}, \mathbf{u})$ that appears in the input-output realization (4.15) of the plant dynamics, represents a mapping $\mathcal{R}^n \times \mathcal{R}^m \rightarrow \mathcal{R}^m$, where $\dim(\mathbf{x}) = n$, $\dim(\mathbf{u}) = m$, and $\dim(\mathbf{y}) = m$. Alternative model mappings can be synthesized that are compatible with the state-space realization (4.2), although in such cases the dimension of the mappings (two are required, in general) is larger: the system state update equation would be augmented with a mapping of the form $\mathcal{R}^n \times \mathcal{R}^m \rightarrow \mathcal{R}^n$, while the output equation would be augmented with a mapping $\mathcal{R}^n \rightarrow \mathcal{R}^m$. One advantage of the input-output realization is an economical use of network resources.

In much of our work, we have used a simple *incremental* gradient learning algorithm [Baker & Farrell (1992)] to update the adjustable parameters \mathbf{p} in the network model $\mathbf{n}(\mathbf{x}, \mathbf{u}; \mathbf{p})$. To employ this particular algorithm, it must be possible to compute (or estimate) the gradient $(\partial J / \partial \mathbf{p})$ of a cost function J , with respect to the parameters \mathbf{p} . The cost function we have chosen to minimize is based on the norm of the output error of the network mapping (e.g., $J = \frac{1}{2} \tilde{\mathbf{n}}^T \tilde{\mathbf{n}}$); thus, an expression for this error is needed. From (4.15) it is easily seen that the output error $\tilde{\mathbf{n}}$ associated with $\mathbf{n}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}; \mathbf{p})$ is given by

$$\tilde{\mathbf{n}} = \mathbf{y}_k - \mathbf{C}\Phi\mathbf{x}_{k-1} - \mathbf{C}\Gamma\mathbf{u}_{k-1} - \mathbf{n}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}; \mathbf{p}) \quad (4.19)$$

Once this error has been determined, the cost J associated with it can be assessed. Consequently, all the information needed to update the adjustable parameters (at any time $t \geq k$) by means of the incremental learning algorithm is contained in the vector-tuple $\{\tilde{\mathbf{n}}, \mathbf{x}_{k-1}, \mathbf{u}_{k-1}\}$. In this setting, training need not be "synchronized," in the sense that the update of the network parameters need not occur immediately, and can in fact occur at a much later point in time, provided that the appropriate vector-tuple of information can be recalled.

A more elaborate discussion of the learning system used in our work is presented in Section 5.6.

4.2 Applications of Hybrid Control to Nonlinear Systems

The indirect hybrid adaptive/learning control methodology outlined in the previous section has been successfully applied to a number of nonlinear dynamical systems. The results associated with three different applications of hybrid control are summarized below. Specific details of each application example and experimental setup are fully documented in the attachments and will not be repeated here.

Each example conforms to the following basic scenario:

- the dynamical system to be controlled is nonlinear
- the only *a priori* design information available about the plant is a single, constant parameter linear model; thus, there is always model uncertainty
- the desired reference model is represented by a stable linear system

- the desired tracking error dynamics are represented by a stable linear system.

4.2.1 Cart-Pole System Split-Level Track Problem

This work is fully documented in §5, pp. 65-100, of Attachment 1 [Baird (1991)]; for related work, see also [Baird & Baker (1990); Baker & Farrell (1990)].¹

The "split-level track problem" is based on the infamous cart-pole system (an inverted pendulum on a translating cart—see figure, p. 65, Attachment 1). The cart-pole problem is a staple of control theory textbooks and learning control papers (e.g., [Kailath (1980); Barto, Sutton, & Anderson (1983); Friedland (1986); Anderson (1989); Morgan, Patterson, & Klopff (1990); Baird & Baker (1990); Baker & Farrell (1990)]). The problem is to move the cart to some desired track position by applying force directly to the cart center of mass, while at the same time balancing a rigid pole that is attached to the cart via a hinge. A key feature of the split-level track problem is that the track is not flat, and instead contains an incline that is not included in the design model.

The hybrid adaptive/learning control methodology, as outlined in Section 4.1, was used as a position controller for the cart-pole system on the split-level track. In a manner consistent with (4.1)-(4.4), the only *a priori* knowledge of the plant that was available was a single linear model. The desired reference model was linear, as was the desired tracking error dynamics.

The nonlinear equations-of-motion for the cart-pole system, open-loop dynamics, model parameter values, definition of the split-level track problem, linearized *a priori* design model, and linear reference model are all discussed in §5.1 of Attachment 1. Experimental results with and without sensor noise are provided in §5.2-§5.6.

¹ The symbol "§" will be used exclusively to refer to sections that appear in the attachments; the word "section" will be used when referring to material that appears in the main document.

Overall, the hybrid controller outperformed both a linear compensator of the form given by (4.9) and an adaptive compensator of the form given by (4.14).¹ In related work, unmodeled actuator dynamics, sensor noise, and a pure time-delay were also incorporated into the split-level track problem—again, the hybrid controller outperformed both the linear and adaptive compensators [Baker & Farrell (1990)].

4.2.2 Aeroelastic Oscillator

This work is fully documented in §4.1, pp. 48-65, of Attachment 2 [Nistler (1992)]; for related work, see also [Atkins (1993); Cerrato (1993)].

The aeroelastic oscillator [Parkinson & Smith (1963)] is a complex nonlinear system that exhibits limit cycle behavior. This system can be represented as a simple, two-state, mass-spring-dashpot model of an aerodynamically driven oscillator, and can also be considered as a simple model of wing flutter or other similar aeroelastic behavior.

The nonlinear equations-of-motion for the aeroelastic oscillator, its open-loop dynamics, linearized *a priori* design model, linear reference model, and the application of the hybrid controller are all discussed on pp. 48-54 of Attachment 2. Two simulation experiments using the hybrid controller were performed. In the first example, the objective was simply to regulate the oscillator to zero position and zero velocity (i.e., to null out limit cycles induced by the nonlinear aerodynamics via an applied force). In the second example, the goal was to command and hold an arbitrary position (deflection), while maintaining zero velocity.

In both experiments, the hybrid adaptive/learning control system outperformed a similar control system having adaptive augmentation only. These results, to-

¹ The hybrid control law (4.18) was successful in this application even though the cart-pole system has unstable zero dynamics. The explanation for this apparent inconsistency is that the reference model used in this work was derived by linearizing the cart-pole system dynamics (about its unstable equilibrium position) and then combining this open-loop model with a stabilizing linear feedback control law. Since the zeros of a transfer function are unaffected by state feedback, the desired reference model included the original nonminimum phase zeros of the plant. As a result, no attempt was made to automatically "cancel" the dynamics associated with these nonminimum phase zeros, and hence the control law was not susceptible to this form of instability.

gether with a three-dimensional plot of the nonlinear dynamics that were synthesized for the learning system, are presented on pp. 55-65 of Attachment 2.

4.2.3 Three-Degree-of-Freedom Flight Control

This work is fully documented in §4.2, pp. 66-96, of Attachment 2 [Nistler (1992)]; for related work, see also [Baker & Millington (1992); Millington & Baker (1992); Baker & Millington (1993); Millington, Baker, & Koenig (1993)].

As discussed in Chapter 3, there are a number of difficulties associated with the design of flight control systems. A three-degree-of-freedom (3-DOF) nonlinear model of the longitudinal dynamics of a representative high performance aircraft was obtained from a full 6-DOF nonlinear aircraft model [Brumbaugh (1991)].¹ This nonlinear 3-DOF model was then used as the basis of a more challenging problem for the hybrid control methodology, relative to the two applications considered above.

A description of the nonlinear aircraft model, linearized *a priori* design model, linear reference model, and a discussion of relevant application issues are all presented on pp. 66-77 of Attachment 2. Once again, two simulation experiments with the hybrid controller were performed. In the first example, the objective of the hybrid controller was to serve as a simple autopilot (i.e., to maintain commanded altitude and airspeed via the use of a horizontal stabilator and throttle). In the second example, the operational envelope for the autopilot was expanded, to make the problem even more challenging.

Once again, the hybrid adaptive/learning control system outperformed a similar control system having adaptive augmentation only. These results are presented on pp. 78-96 of Attachment 2.

4.3 Learning Augmented Estimation

The estimation or reconstruction of system state variables from observed output measurements typically requires an accurate model of the system. On the other

¹ This 6-DOF model is identical to that used in the AIAA Controls Design Challenge.

hand, the hybrid control and learning schemes outlined in Section 4.1 assumed the *absence* of an accurate model, but the *availability* of accurate state estimates. Thus, a more general problem exists, involving simultaneous system identification, state estimation, and control. Obviously, learning could be used to address some aspects of this more general problem if the estimation and control processes were allowed to utilize the modeling information provided by a learning system. This is the basic idea underlying *learning augmented estimation*. The material presented in this section only represents a first step in this direction—there is much room for further analysis and development.

Beginning with a standard linear state estimator, the discussion below proceeds by successively incorporating adaptation and learning in manner completely parallel to the development of the hybrid adaptive/learning control methodology.

Linear Estimation

The standard, steady-state, Kalman filter propagation and update equations for a linear dynamical system of the form (4.2) are given by ([Kalman (1960)]; see also [Friedland (1986); Gelb (1974); Jazwinski (1970); Maybeck (1979); Sorenson (1985)]):

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \Phi \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_{k-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{y}_k^m - \mathbf{C} \hat{\mathbf{x}}_k^-)\end{aligned}\tag{4.20}$$

where $\hat{\mathbf{x}}_k^-$ indicates the estimate of the state after propagation, but prior to incorporation of the measurement \mathbf{y}_k^m , while $\hat{\mathbf{x}}_k$ represents the final state estimate after the measurement update. The matrices Φ , Γ , and \mathbf{C} are given by the assumed model of the system, and \mathbf{K} is the steady-state Kalman gain matrix (which is the optimal gain matrix under certain mild assumptions).

If the actual system dynamics differ from those described by (4.2) (e.g., if the system is nonlinear), then the estimates generated by (4.20) will be suboptimal, and in fact may be quite inaccurate.

Adaptive Augmentation

In an attempt to accommodate such modeling error in the estimation process, we can add an *adaptive* component to the estimator. This component seeks to im-

prove the accuracy of the propagation equation by addressing the *modeling error* of the process, using the difference between the final state estimate at the previous step and the previous linear propagation as a kind of *time-delay estimate* (in a fashion completely analogous to what was done previously in Section 4.1). In this case the design model for the process becomes:

$$\begin{aligned}\mathbf{x}_k &= \Phi \mathbf{x}_{k-1} + \Gamma \mathbf{u}_{k-1} + \xi_{k-1} \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k\end{aligned}\quad (4.21)$$

where the vector ξ represents the modeling error in the state equation (for the present, we assume that there is no modeling error in the output equation). A time-delay estimate for ξ_{k-1} can be obtained from previous state estimates as follows:

$$\hat{\xi}_{k-1} = \hat{\mathbf{x}}_{k-1} - \Phi \hat{\mathbf{x}}_{k-2} - \Gamma \mathbf{u}_{k-2}$$

Based on this estimate of the unmodeled dynamics, a new propagation equation can be written that incorporates such adaptation

$$\hat{\mathbf{x}}_k^- = \Phi \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_{k-1} + \hat{\xi}_{k-1}$$

A complete set of adaptive filter equations can be defined using this propagation equation and the update equation from (4.20):

$$\begin{aligned}\hat{\xi}_{k-1} &= \hat{\mathbf{x}}_{k-1} - \Phi \hat{\mathbf{x}}_{k-2} - \Gamma \mathbf{u}_{k-2} \\ \hat{\mathbf{x}}_k^- &= \Phi \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_{k-1} + \hat{\xi}_{k-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{y}_k^m - \mathbf{C} \hat{\mathbf{x}}_k^-)\end{aligned}\quad (4.22)$$

These equations can also be rewritten in a form that is similar to that of (4.20); when this is done, one can see that this scheme introduces an intermediate *adaptive step*, resulting in the following three-step process:

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= \Phi \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_{k-1} \\ \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + (\hat{\mathbf{x}}_{k-1} - \hat{\mathbf{x}}_{k-1}^-) \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^+ + \mathbf{K}(\mathbf{y}_k^m - \mathbf{C} \hat{\mathbf{x}}_k^+)\end{aligned}$$

where in this case $\hat{\mathbf{x}}_k^-$ represents the state perturbation estimate *after* propagation *but prior* to the adaptive correction and measurement update, $\hat{\mathbf{x}}_k^+$ represents the state estimate after the adaptive correction is made, and $\hat{\mathbf{x}}_k$ is the final state estimate. Some empirical results [Millington & Baker (1992)] have verified the ability of this *adaptive augmentation* to address estimation difficulties introduced by unmodeled nonlinearities.

Note that this approach does not account for the possibility of modeling error in the output equation. If a new design model for the process were used

$$\begin{aligned}\mathbf{x}_k &= \Phi \mathbf{x}_{k-1} + \Gamma \mathbf{u}_{k-1} + \xi_{k-1} \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k + \omega_k\end{aligned}\quad (4.23)$$

then one could obtain time-delay estimates for the modeling errors ξ_{k-1} and ω_k , and ultimately arrive at the following adaptive filter equations:

$$\begin{aligned}\hat{\xi}_{k-1} &= \hat{\mathbf{x}}_{k-1} - \Phi \hat{\mathbf{x}}_{k-2} - \Gamma \mathbf{u}_{k-2} \\ \hat{\omega}_k &= \mathbf{y}_{k-1}^m - \mathbf{C} \hat{\mathbf{x}}_{k-1} \\ \hat{\mathbf{x}}_k^- &= \Phi \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_{k-1} + \hat{\xi}_{k-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{y}_k^m - \mathbf{C} \hat{\mathbf{x}}_k^- - \hat{\omega}_k)\end{aligned}\quad (4.24)$$

Learning Augmentation

Like adaptation, learning can also be incorporated into the estimation process. Following the same development procedure as before, we begin with a design model for the process:

$$\begin{aligned}\mathbf{x}_k &= \Phi \mathbf{x}_{k-1} + \Gamma \mathbf{u}_{k-1} + \mathbf{n}^x(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \xi_{k-1} \\ \mathbf{y}_k &= \mathbf{C} \mathbf{x}_k + \mathbf{n}^y(\mathbf{x}_k) + \omega_k\end{aligned}\quad (4.25)$$

where \mathbf{n}^x and \mathbf{n}^y represent two distinct mappings that will be synthesized by the learning system, and ξ and ω represent any residual unmodeled dynamics. As before, both \mathbf{n}^x and \mathbf{n}^y are implicitly functions of time since they will be evolving due to learning.

Given the design model (4.25), the complete set of hybrid adaptive/learning augmented filter equations becomes

$$\begin{aligned}\hat{\xi}_{k-1} &= \hat{\mathbf{x}}_{k-1} - \Phi \hat{\mathbf{x}}_{k-2} - \Gamma \mathbf{u}_{k-2} - \mathbf{n}^x(\hat{\mathbf{x}}_{k-2}, \mathbf{u}_{k-2}) \\ \hat{\omega}_k &= \mathbf{y}_{k-1}^m - \mathbf{C} \hat{\mathbf{x}}_{k-1} - \mathbf{n}^y(\hat{\mathbf{x}}_{k-1}) \\ \hat{\mathbf{x}}_k^- &= \Phi \hat{\mathbf{x}}_{k-1} + \Gamma \mathbf{u}_{k-1} + \mathbf{n}^x(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) + \hat{\xi}_{k-1} \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{y}_k^m - \mathbf{C} \hat{\mathbf{x}}_k^- - \mathbf{n}^y(\hat{\mathbf{x}}_k^-) - \hat{\omega}_k)\end{aligned}\quad (4.26)$$

An incremental gradient learning algorithm can be used to update each of the required mappings. In this case, the network output errors $\tilde{\mathbf{n}}^x$ and $\tilde{\mathbf{n}}^y$ associated with $\mathbf{n}^x(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1})$ and $\mathbf{n}^y(\hat{\mathbf{x}}_k)$, respectively, are given by:

$$\begin{aligned}\bar{\mathbf{n}}^x &= \hat{\mathbf{x}}_k - \Phi \hat{\mathbf{x}}_{k-1} - \Gamma \mathbf{u}_{k-1} - \mathbf{n}^x(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) \\ \bar{\mathbf{n}}^y &= \mathbf{y}_k^m - \mathbf{C} \hat{\mathbf{x}}_k - \mathbf{n}^y(\hat{\mathbf{x}}_k)\end{aligned}$$

Additional Remarks

In general and for best results, the gain matrix \mathbf{K} should be chosen to reflect the local (linearized) system dynamics and, hence, should be adjusted on-line. One means for determining a suitable gain matrix is to compute the optimal steady-state Kalman gain matrix associated with the local (linearized) system.

Note also that this approach requires that *two* mappings (one for the state equation and one for the output equation) be synthesized via learning. Because our other work with the hybrid adaptive/learning control methodology was based on the collapsed input/output system description given by (4.5) (which involved a single mapping), we did not have the opportunity to evaluate the performance of (4.22), although we fully expect it to perform better than either (4.20) or (4.21) when there unmodeled nonlinearities are present.

5 Multiaxis Flight Control

A main objective of this program was to demonstrate the feasibility and advantages of learning augmented flight control in the context of six-degree-of-freedom (6-DOF), multiaxis flight. The specific feature to be demonstrated was the ability of a learning augmented flight control system to provide high performance control despite significant modeling uncertainty and nonlinearities in the aircraft dynamics. Accordingly, a hybrid adaptive/learning control system was developed and applied to a challenging multiaxis flight control problem. The performance of the resulting learning augmented flight controller was contrasted with three similar control systems: an unaugmented compensator (*a priori* linear design only), an adaptively augmented compensator, and a compensator having essentially "ideal learning" augmentation.

Of the many specific aircraft control problems that could be selected for this study, it was desired to select one that would minimize unnecessary engineering complexity while still demonstrating the advantages of learning augmented flight control. To this end, a longitudinal and lateral/directional control augmentation system (CAS) with turn coordination was used as the target control application. The rationale for this selection and further details of the problem are discussed in Section 5.1. The hybrid controller was evaluated via a closed-loop simulation of a 6-DOF, nonlinear aircraft model [Brumbaugh (1991)]. A brief overview of this model is given in Section 5.2. The salient characteristics of the open-loop dynamics of the vehicle are presented in Section 5.3. The "invertibility" of the assumed plant dynamics will be of particular interest, given the nonlinear control techniques used in this study.

There are numerous ways in which learning might be applied to the selected flight control problem. Three such methods are presented in Section 5.4. A qualitative evaluation of the candidate methods leads to the selection of a variant of the hybrid learning/adaptive approach which was discussed in Chapter 4. The application of this scheme to the 6-DOF CAS design for the nonlinear aircraft model is presented in Section 5.5 and includes a description of the learning system which was used. Finally, Section 5.6 presents experimental results of the hybrid control system in the context of a challenging "S trajectory" maneuver.

5.1 Control Problem Definition

There are a wide range of 6-DOF flight control problems that can be studied. These run the gamut from altering the natural modes of the aircraft; i.e., stability augmentation systems, to the higher level functions of control augmentation systems and autopilot design. Under some design procedures the higher level functions encompass lower level designs. The desire to minimize engineering complexity and avoid man-in-the-loop issues, while still achieving the task objective, has led to the selection of a control augmentation system as the design problem. Specifically, the demonstration problem is the design of a longitudinal and lateral/directional CAS with turn coordination.

The demonstration platform is a nonlinear model of a modified F-15, as discussed in [Brumbaugh (1991)]. The dynamics of this vehicle are inherently nonlinear in angle-of-attack α , sideslip β , airspeed V , and altitude H . This aircraft has four control inputs available: rudder δ_r , aileron δ_a , symmetric stabilator δ_h , and differential stabilator δ_d . The throttle setting is assumed to be independently controlled (we held it at a constant value). Thus, *the control problem is to use the four control inputs to track reference values of the stability axis pitch rate q_c and roll rate p_c , while regulating β to zero, as shown in Fig. 5.1.*

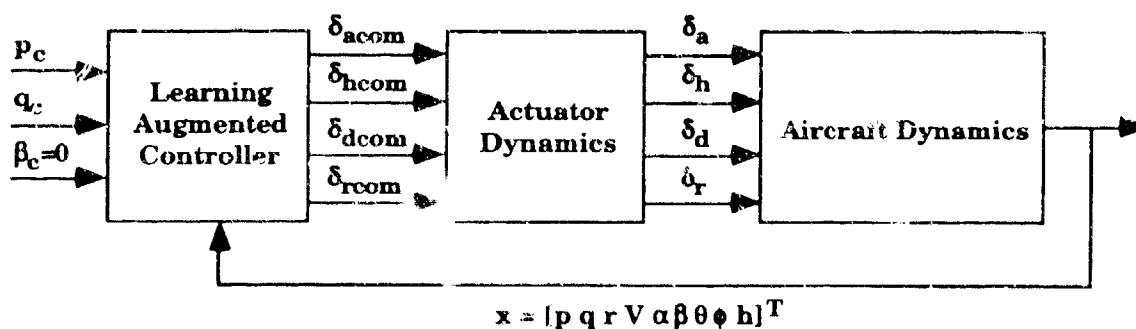


Figure 5.1. The Basic Control Problem.

For the purposes of this study, it is not necessary to design a full-envelope controller. The intent is to design and demonstrate a controller that will operate over a sufficiently large region of the envelope so as to exercise the nonlinear dynamics of the system. To that end, the demonstration will focus on a single demanding

"S-trajectory" maneuver involving rapid airspeed variation, high angles-of-attack, and significant coupling between the different degrees of freedom.

5.2 Flight Simulation

The nonlinear aircraft model, including its actuators, sensors, and atmospheric environment, are briefly outlined in this section. For more information regarding this particular vehicle model, please refer to pp. 66-74 of Attachment 2, or to [Brumbaugh (1991)].¹

Aircraft Model

The simulation code used in this work is based on a six-degree-of-freedom, rigid-body, high performance aircraft model, including nonlinear aerodynamic effects (based on empirically derived tabular data), nonlinear engine dynamics, and nonlinear actuator dynamics (including rate and position limits). The control features of this aircraft include: (i) a rudder surface mounted on a single vertical tail, (ii) an all moving horizontal tail (stabilator) capable of symmetric and differential movement, and (iii) wing ailerons. A more detailed description of the basic aircraft model can be found in [Brumbaugh (1991)].

Actuator Models

All control surfaces employ identical actuator dynamics, with 0.033 s time-constants and 35°/s rate limiting. Additionally, the stabilator is constrained by asymmetric position limits of +15° / -25°, while the aileron and rudder saturate symmetrically at $\pm 20^\circ$ and $\pm 30^\circ$, respectively.

Sensor Models

We assume the controller has access to a full-state sensor suite, including: wind relative angles (angle-of-attack and sideslip angles), altimeter, airspeed and Mach indicators, as well as roll, pitch, and heading attitudes. In this work, the sensors were assumed to be ideal (i.e., to contribute no noise or delay).

¹ This 6-DOF model is identical to that used in the *AIAA Controls Design Challenge*.

Gust Model

To demonstrate the robustness properties of our system while it was learning, we incorporated moderate atmospheric turbulence into our simulation. Specifically, we used a Dryden gust model [MIL-STD-1797A (1990)].

5.3 Open-Loop Dynamics

Before proceeding with any type of control system design—whether conventional, adaptive, learning, or whatever—it is important to examine any *a priori* information about the plant dynamics which may be available. At the very least, this information guides the design of the overall control system architecture and, in our approach, provides the required information to develop the fixed, conventional component of the hybrid controller.

The aircraft model consists of a set of first-order, nonlinear differential equations [Brumbaugh (1991)]. Many of the "coefficients" of these nonlinear equations are nonlinear functions of angle-of-attack, airspeed, sideslip angle, altitude, and the applied controls. To characterize the dynamics of this aircraft, the nonlinear dynamics were numerically linearized at several equilibrium points throughout the operating envelope, ranging in speed from 600 to 1,000 ft/s and in altitude from 5,000 to 40,000 ft. The aircraft eigenvalues and eigenvectors were found to be quite typical of an aircraft of this type. The aircraft is open-loop stable with the exception of an unstable phugoid mode at a low altitude, high speed operating point (5,000 ft, 987 ft/sec). The operating regime near $H = 5,000$ ft and $V = 600$ ft/s is of particular interest since the evaluation maneuvers are initiated from this point. The modal frequencies are shown in Table 5.1 below.

Table 5.1. Modal Frequencies for $H = 5,000$ ft and $V = 600$ ft/s.

| Frequency (rad/s) | Mode |
|---------------------|------------------|
| $-1.79 \pm 2.51i$ | short period |
| $-0.48 \pm 3.01i$ | dutch roll |
| -2.78 | roll convergence |
| -1.60 | engine core |
| $-0.015 \pm 0.076i$ | phugoid |
| -0.026 | spiral |
| -0.002 | altitude |
| 0.0 | heading |

Since many of the candidate control methodologies perform some form of nonlinear inversion of the plant dynamics, the zero dynamics (minimum/nonminimum phase) characteristics of the plant are of particular interest. If the tracking outputs are chosen to be $y = [p, q, \beta]^T$, then the system has a multivariable nonminimum phase zero at 0.003 rad/s. There is also a nonminimum phase zero in the single-input/single-output (SISO) transfer function $\beta(s)/\delta_r(s)$ at 0.043 rad/s. The consequence of these nonminimum phase properties is that any attempt to control sideslip β directly via the rudder input δ_r , using linear or nonlinear inversion techniques will likely result in an unstable closed-loop system.

5.4 Control Methodology

Conventionally, the control system "architecture" refers to the specification of the measurements and controls, and the feedback paths between these variables. The control objectives, the open-loop dynamics of the aircraft, and the availability and complexity of the design methodologies are the three most important factors in determining the appropriate control architecture in conventional control schemes. For learning augmented controllers, the augmentation mechanism is an additional design degree-of-freedom that must be addressed.

5.4.1 Candidate Methodologies

In this subsection, three different learning augmentation schemes are presented and evaluated in the context of the CAS problem.

Feedforward Augmentation

The feedforward learning augmentation concept (see Fig. 5.2) is based on the so-called "two-parameter" compensator [Vidyasagar (1985)]. As the name implies the two-parameter compensator consists of two components—a feedback compensator $K_f(s)$ which provides robust stability and disturbance rejection, and a feedforward compensator (or prefilter) $K_{ff}(s)$ which is tuned for tracking performance. In the feedforward learning augmentation scheme, the feedback compensator is determined by standard robust linear design techniques. However, the feedforward compensator is designed on-line, by using the learned local linearized dynamics of the inner-loop (plant plus feedback compensator) throughout the flight envelope. During the early stages of learning (prior to parameter convergence), the linear model of the inner-loop might be identified via a conventional recursive identification scheme.

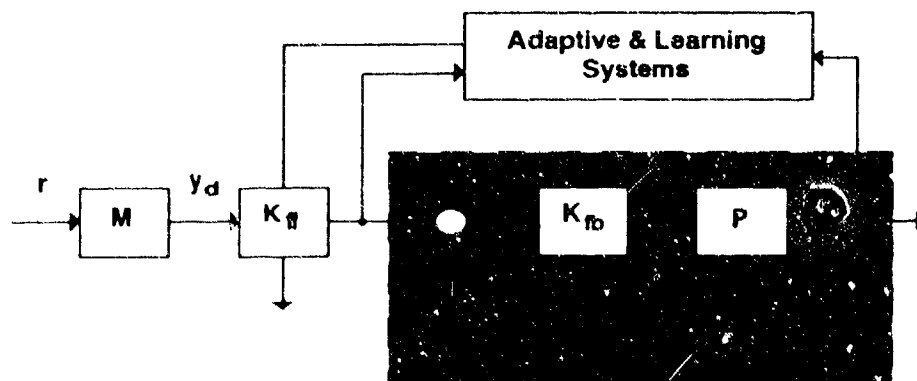


Figure 5.2. Feedforward Learning Augmentation.

The on-line design procedure involves the stable inversion of the learned, linearized model. Given knowledge of the local linearized inner-loop dynamics $P_{aug}(s)$, the prefilter K_{ff} that minimizes the H_2 norm [Maciejowski (1989)] of the tracking error

$$e = y - y_d = (I - P_{aug} K_{ff}) y_d$$

is the stable inverse of P_{aug}

$$K_{ff} = P_o^{-1} (P_i^{-1})_s$$

where $P_{aug} = P_i P_o$ is the inner/outer factorization [Maciejowski (1989)] of P_{aug} , and $(P_i^{-1})_s$ denotes the stable part of P_i^{-1} . This technique is tolerant of nonminimum phase dynamics since inversion of only the minimum phase portion of the plant ensures that K_{ff} is a stable transfer function.

The advantage of this architecture is that the features of learning augmentation will enhance the performance of the system in the presence of modeling uncertainties, unexpected changes in the plant dynamics, and nonlinearities, and yet the stability of the inner-loop will be insensitive to learning failures if the learning dynamics are slow relative to the inner-loop dynamics (which is normally the case). This "learning-fail-safe" architecture might be appropriate for flight test applications, where reliability is critical.

One disadvantage of this scheme is the requirement to identify the local linear representation of the augmented plant. The identification of such a highly structured model poses problems for both learning based and conventional system identification algorithms. While the current learning algorithm (see Section 5.6) provides a good input/output map of the desired nonlinear function, extracting local linear models from the map can be difficult and can exacerbate small errors due to the fact that one must differentiate this map to find the local Jacobian matrices with respect to the inputs. Additionally, the development of a system identification algorithm (e.g., via an extended Kalman filter or recursive maximum likelihood technique) for 6-DOF aircraft dynamics is an enormous problem in its own right. Substantial supervisory logic could be required to guarantee parameter convergence in the presence of disturbances and sensor noise.

The computational burden associated with this on-line inversion algorithm is another concern. Let n be the number of states of the system. The stable inversion algorithm involves the solution of the algebraic matrix Riccati equation [Kailath (1980)], which requires on the order of $73n^2$ multiplications/divisions, $82n^3$ additions/subtractions, and $3n^2$ square root operations when using the Schur method [Ramesh, Senol, & Garba (1989)].

Plant Augmentation

This scheme is an extension of the model reference adaptive control (MRAC) concept [Åström & Wittenmark (1989); Narendra & Annaswamy (1989)]. In this case, the controller would consist of a linear reference model $P_m(s)$, the adaptive/learning system, and one of two linear compensators, $K_{rob}(s)$ and $K_{hp}(s)$. The purpose of the adaptive and learning systems is to augment the dynamics of the nonlinear plant so that it has the same input/output behavior as the reference model. The usual division of responsibilities between the adaptive and learning components applies here, with the adaptive component accommodating time-varying dynamics and the learning component addressing state-dependent nonlinearities. The reference model $P_m(s)$ is a linearized model of the nominal plant dynamics (see Fig. 5.3).

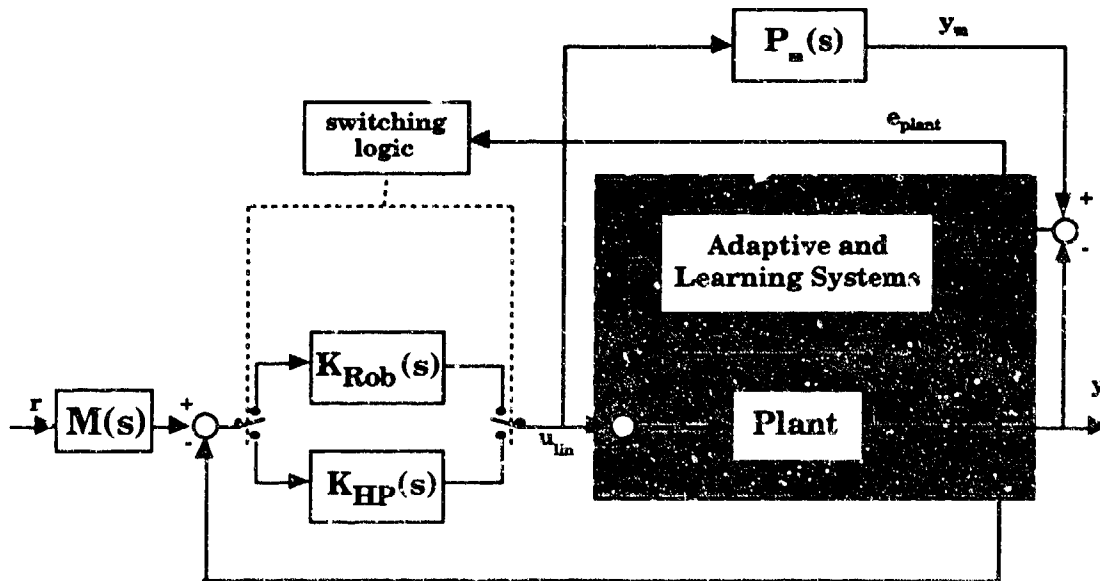


Figure 5.3. Plant Augmentation.

At any instant, linear control is provided by either a "robust" compensator $K_{rob}(s)$, or a "high performance" compensator $K_{hp}(s)$. The robust controller would be used during the early stages of learning when the augmented plant (shaded area of Fig. 5.3) may deviate significantly from $P_m(s)$. During this period, the large plant residual e_{plant} will cause the switching logic to close the loop through the robust controller. When learning has converged, the switching logic closes the

outer-loop through the high performance controller. Both linear controllers would be synthesized using the H_∞ , μ synthesis robust design methodology; however, the modeling uncertainty $\Delta(s)$ assumed in the "robust" design would be much greater than that assumed for the "high performance" design (see Fig. 5.4). Thus, $K_{rob}(s)$ should be more robust to errors in the augmentation process, while $K_{hp}(s)$ should be more finely tuned to the reference model $\tilde{M}(s)$, and should provide superior performance when the augmented dynamics are close to $P_m(s)$.

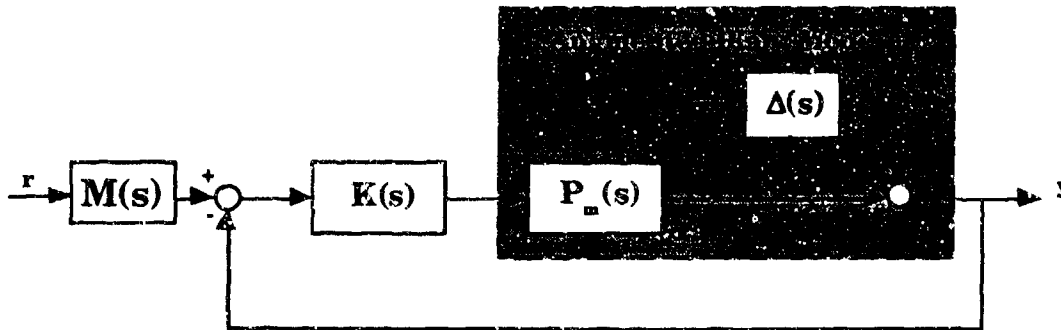


Figure 5.4. Model for Linear Compensator Design.

The residual monitoring and subsequent transition to the robust compensator provides for a limited fail-safe quality if the learning and/or adaptation algorithms fail to provide the desired augmentation. Performance would be excellent when the learning/adaptation systems have converged.

Augmentation Via Dynamic Inversion

Dynamic inversion has been used in the robotics field and in many other applications for a number of years [Slotine & Li (1991)]. It is a highly effective scheme when the plant model is well known. A major weakness of this scheme is the high degree of performance sensitivity to modeling error. This is an area where adaptation and learning augmentation could provide a significant performance boost. Adaptive and learning systems (see Fig. 5.5) could jointly compute an estimate $\hat{f}(\mathbf{x}, \mathbf{u})$ of the actual plant dynamics $f(\mathbf{x}, \mathbf{u})$. A control would be selected that directly cancels the nonlinear dynamics (as characterized by the estimate) and then injects the desired error dynamics. This is essentially the same hybrid control scheme that was detailed in Section 4.1. A brief summary of this scheme as it might apply to the attitude rate control problem is presented below.

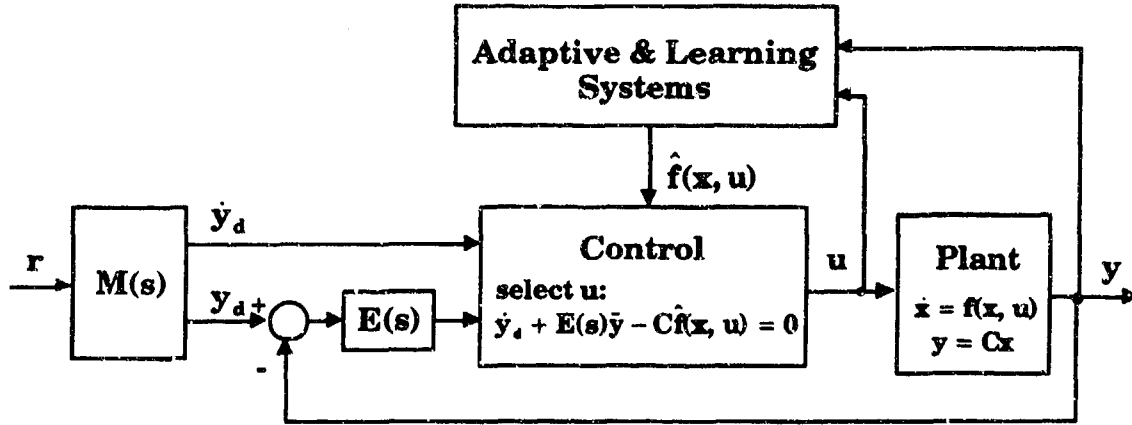


Figure 5.5. Augmentation via Dynamic Inversion.

Assume that the plant dynamics and outputs are given in continuous time by

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{C}\mathbf{x}\end{aligned}$$

and the desired tracking error dynamics are specified as

$$\dot{\tilde{\mathbf{y}}} + \mathbf{E}\tilde{\mathbf{y}} = \mathbf{0}$$

where $\tilde{\mathbf{y}} = \mathbf{y}_d - \mathbf{y}$, and \mathbf{E} is a diagonal matrix with positive elements (so the system is stable). If the error dynamics are enforced, the output errors \tilde{y}_i will exponentially decay to zero with time constants $\tau_i = 1/E_{ii}$. The error dynamics may be expressed in terms of the plant dynamics by differentiating the output equation

$$\begin{aligned}\dot{\mathbf{y}} &= \mathbf{C}\dot{\mathbf{x}} \\ &= \mathbf{C}\mathbf{f}(\mathbf{x}, \mathbf{u})\end{aligned}$$

and substituting this into an expression for the derivative of the error

$$\begin{aligned}\dot{\tilde{\mathbf{y}}} &= \dot{\mathbf{y}}_d - \dot{\mathbf{y}} \\ &= \dot{\mathbf{y}}_d - \mathbf{C}\mathbf{f}(\mathbf{x}, \mathbf{u})\end{aligned}$$

The hybrid adaptive/learning system can be used to generate an estimate of the plant dynamics $\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u})$. By selecting \mathbf{u} such that

$$\dot{\mathbf{y}}_d + \mathbf{E}\tilde{\mathbf{y}} - \mathbf{C}\hat{\mathbf{f}}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$$

the desired error dynamics are obtained if the plant is identified exactly.

The key to this approach is the accurate estimation of the plant dynamics. Also, caution must be used in selecting the plant inputs and outputs to ensure that the

plant dynamics are minimum phase, since any attempt to invert nonminimum phase dynamics may lead to instability.

This scheme is attractive because only an input/output representation of the dynamics is required—a structured model is not necessary. In addition, this approach does not require the development of a structured adaptive control or system identification algorithm. The simple time-delay adaptive augmentation described in Section 4.1 is a much simpler adaptive algorithm, and may be used to facilitate learning during its convergence phase and to handle novel and time-varying dynamics.

5.4.2 Control Methodology Selection

Both the feedforward augmentation and plant augmentation schemes offer some interesting features. For example, both have some degree of robustness to learning and adaptation imperfections. Unfortunately, these schemes require significant development of enabling technologies before they can be applied to the problem at hand. The dynamic inversion algorithm is a simpler, proven algorithm that has been demonstrated on aircraft control problems of a smaller scale (e.g., see Section 4.2). Given the limited scope of the current effort and the past experience gained with this approach, the learning augmented dynamic inversion scheme was selected as the control methodology to be used in conjunction with the 6-DOF CAS design demonstration.

5.5 Controller Design

This section describes the design of the control augmentation system using the hybrid control scheme detailed in Chapter 4. In review, the objective of the control system is to track pitch rate q_c and roll rate p_c commands (in stability axes), while maintaining turn coordinated flight ($\beta_c = 0$). This section begins with a description of the top-level architecture, followed by a detailed description of the *a priori*, adaptive, and learning components.

5.5.1 Control Architecture

The control system consists of an outer-loop β regulator and an inner-loop angular rate tracker (see Fig. 5.6). The inputs are commanded stability-axis roll rate p_c , pitch rate q_c , and sideslip angle $\beta_c = 0$. This type of structure with the attitude rates controlled via dynamic inversion in an inner-loop, and the wind-axis attitudes controlled via dynamic inversion in an outer-loop, has been used successfully in other studies (e.g., [Bugajski, Enns, & Elgersma (1990)]). The outer β loop is necessary because it makes neither physical nor mathematical sense to regulate the sideslip angle under direct rudder control. The predominant forcing term of the β dynamics is the stability-axis yaw rate. In addition, the dynamics from the rudder command to sideslip angle are nonminimum phase. The consequence of performing dynamic inversion on a nonminimum phase plant is analogous to attempting to cancel a right half-plane zero with a controller pole. Thus, stability-axis yaw rate is used as a pseudo control for the β controller, and this rate command along with the stability-axis roll and pitch rate commands comprise the inputs to the angular rate controller. The following subsections present the details of the fixed, adaptive, and learning components of the angular rate and β controllers.

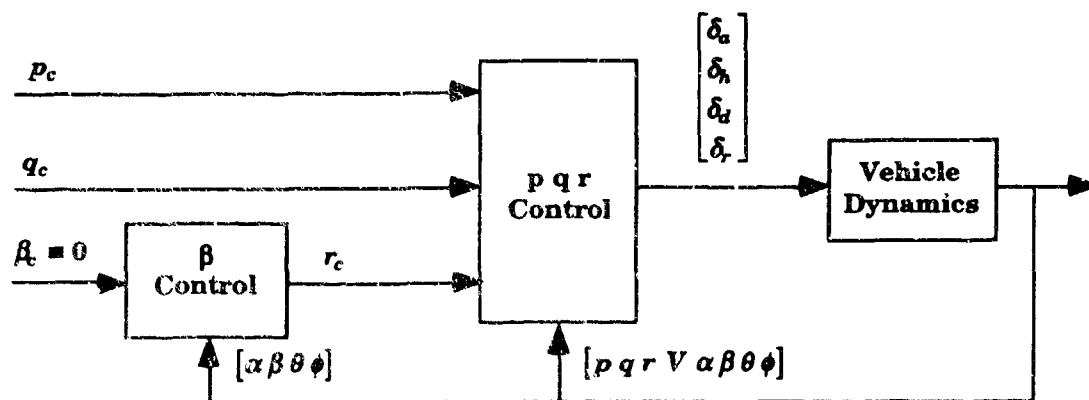


Figure 5.6. Top-level Architecture of the Angular Rate CAS.

5.5.2 Angular Rate Control

The angular rate control system (see Fig. 5.7) is composed of reference model dynamics, error dynamics, and plant inversion functions. The reference model

and error dynamics are performed in stability-axes, while the plant inversion function is performed relative to the body-axis dynamics. The reference model is a prefilter that specifies the desired dynamics of the aircraft under "perfect" control (i.e., if initial tracking errors are zero, there are no disturbances present, and plant inversion is performed exactly). The inputs to the reference model are the commanded stability-axis body rates $[p_c, q_c, r_c]^T$ and the outputs are the reference stability-axis body rates $[p_r, q_r, r_r]^T$. Since disturbances are always present, and the plant inversion process will not be perfect, some error may accumulate. The error dynamics specify how the system is to respond to rate errors. If, for example, the initial tracking error is significant, it is unreasonable to ask the plant to converge to the reference input in one control cycle (deadbeat response), since the required control effort will be excessive. Thus, the error dynamics smooth the tracking convergence phase over several control cycles. The error dynamics may also be augmented with integral action to boost the low frequency gain of the system. The objective of plant inversion is to determine the control vector that drives the system outputs to their desired values $[p_d, q_d, r_d]^T$ over the next control cycle. Plant inversion is performed by using *a priori* information as well as adaptively gained and learned knowledge of the plant dynamics.

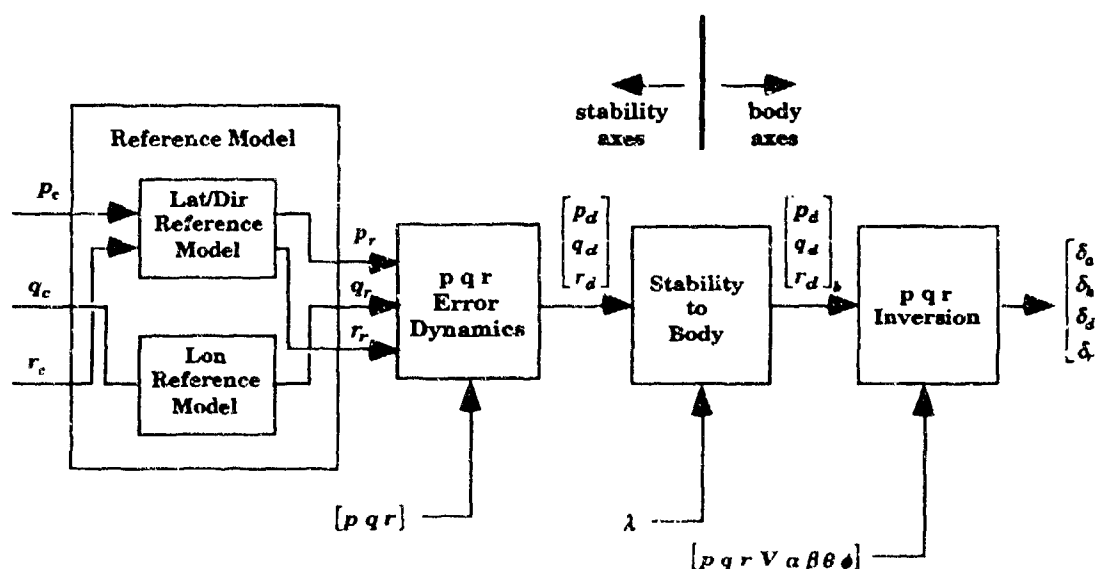


Figure 5.7. The Angular Rate Control System.

Reference Model

The reference model embodies the desired dynamics for the closed-loop system. It also acts as a lowpass filter for any "non-trackable" high frequency signals present in the command inputs, so that the reference signals sent to the controller are physically realizable by the aircraft. Separate lateral/directional and longitudinal reference models simplify the design while ensuring consistency between yaw rate and roll rate reference signals. The reference models were generated by a four-step procedure involving: (i) linearization of the nonlinear equations-of-motion about a nominal operating point at $H = 5,000$ ft and $V = 600$ ft/s; (ii) design of a linear-quadratic (LQ) servo controller, which yielded the desired dynamics; (iii) determination of the closed-loop dynamics of the system under LQ servo control; and (iv) model reduction and conversion to discrete-time of the closed-loop dynamics. The resulting reference models represent compact, linear set of difference equations of the form

$$\mathbf{y}_{r,k+1} = \mathbf{C}_r(\Phi_r \mathbf{x}_{r,k} + \Gamma_r \mathbf{u}_k)$$

where $\mathbf{y}_r = [p_r, r_r]^T$, $\mathbf{u} = [p_c, r_c]^T$, and $\mathbf{x} \in \mathcal{R}^4$ for the lateral/directional model, and $\mathbf{y} = q_r$, $\mathbf{u} = q_c$, and $\mathbf{x} \in \mathcal{R}^4$ for the longitudinal reference model. This procedure guarantees that the reference signals are consistent and achievable (at least in a linear sense).

Error Dynamics

The error dynamics models determine how the system will react to initial tracking errors, imperfect plant inversion, and disturbances. Fast error dynamics provide rapid convergence to the reference values at the expense of a higher level of control effector activity. Slow error dynamics provide sluggish response, but with lower levels of control activity. The p, q, r error dynamics consist of three decoupled difference equations that provide both proportional and integral (PI) compensation of the output errors. The error dynamics (see Fig. 5.8) generate the desired output y_d that is sent to the plant inversion function. The desired output is given by

$$y_{d,k+1} = y_{r,k+1} - k_e e_k - k_i \sigma_k$$

where $e_k = y_{r,k} - y_k$ and σ_k is the integral of the error. If plant inversion is perfect (i.e., if $y_{k+1} = y_{d,k+1}$), then the error dynamics are given by

$$e_{k+1} - k_e e_k - k_i \sigma_k = 0$$

If plant inversion is imperfect, then the error dynamics become

$$e_{k+1} - k_e e_k - k_i \sigma_k = \delta_k$$

where δ_k is an inversion error term; i.e., $y_{k+1} = y_{d,k+1} - \delta_k$. The main feature of the integral term is its ability to drive the output error to zero even in the presence of plant model bias errors.

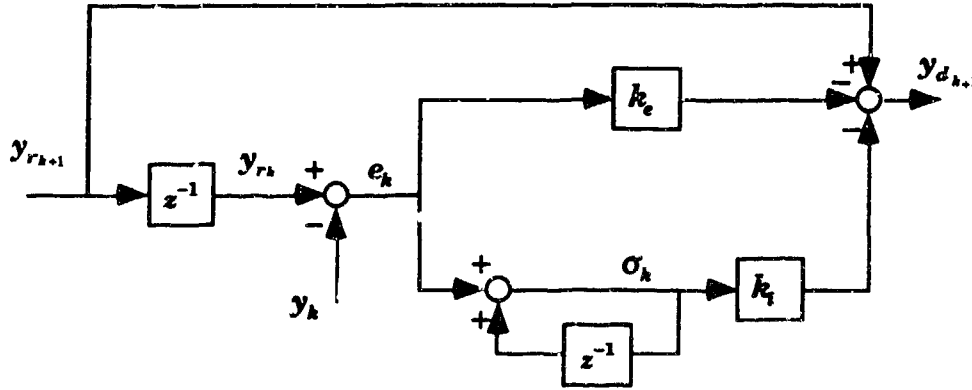


Figure 5.8. Proportional Plus Integral Error Dynamics.

Plant Inversion

The plant inversion process computes the current control u_k as a function of the current state x_k such that the output at the next time step is equal to the desired output $y_{k+1} = y_{d,k+1}$. For the inner rate loop, the control consists of the four control surface commands

$$u_k = [\delta_a \ \delta_h \ \delta_d \ \delta_r]^T_k$$

and the state vector consists of the three body rates, airspeed, angle-of-attack, sideslip angle, and the pitch and roll angles

$$x_k = [p \ q \ r \ V \ \alpha \ \beta \ \theta \ \phi]^T_k$$

and the outputs are the body rates

$$y_k = [p \ q \ r]^T_k$$

The control system is composed of an *a priori* linear component, an adaptive component, and a learning component. By design, the *a priori* model of the plant is a poor representation of the dynamics over the maneuver envelope. This was done to facilitate evaluation of the hybrid learning/adaptive augmentation.

A Priori Rate Control

The linear *a priori* control is determined via dynamic inversion of a linear model of the aircraft. The linear model was obtained by numerically linearizing the nonlinear plant at a trim condition corresponding to an airspeed of 600 ft/s, an altitude of 5,000 ft, and an angle-of-attack of 3 deg. This is a poor model over the spectrum of conditions that will be experienced during the demonstration maneuver, where airspeed, altitude, and angle-of-attack range from 400-1,100 ft/s, 500-23,000 ft, and 0-20 deg, respectively.

Given the nonlinear system $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ with trim condition $\mathbf{x}^o = \mathbf{f}(\mathbf{x}^o, \mathbf{u}^o)$, the linear model may be expressed as

$$\mathbf{x}_{k+1} - \mathbf{x}^o = \Phi(\mathbf{x}_k - \mathbf{x}^o) + \Gamma(\mathbf{u}_k - \mathbf{u}^o) \quad (5.1)$$

$$\mathbf{y}_k - \mathbf{y}^o = \mathbf{C}(\mathbf{x}_k - \mathbf{x}^o) \quad (5.2)$$

The output dynamics are given by substituting (5.1) into (5.2)

$$\mathbf{y}_{k+1} = \mathbf{C}\Phi(\mathbf{x}_k - \mathbf{x}^o) + \mathbf{C}\Gamma(\mathbf{u}_k - \mathbf{u}^o) + \mathbf{y}^o \quad (5.3)$$

The linear control is determined by setting $\mathbf{y}_{d,k+1} = \mathbf{y}_{k+1}$ and solving for \mathbf{u}_k

$$\mathbf{u}_k = \mathbf{u}^o + (\mathbf{C}\Gamma)^+ [\mathbf{y}_{d,k+1} - \mathbf{y}^o - \mathbf{C}\Phi(\mathbf{x}_k - \mathbf{x}^o)] \quad (5.4)$$

where $(\cdot)^+$ denotes the pseudo inverse of the operand. Since $(\mathbf{C}\Gamma)$ is of full rank, a solution always exists and the solution minimizes the Euclidean norm of the control vector.

Adaptive Augmentation

Adaptive control is required to compensate for the nonlinear dynamics while the relatively slow process of learning builds the required input-output map. The type of adaptation selected is related to time-delay control (TDC) [Yousef-Toumi & Ito (1990)]. This is a simple scheme that is easy to implement and is compatible with the formulation of the *a priori* and learning control components. Time-delay control estimates an unstructured forcing term in the dynamics by comparing the predicted output provided by the linear model with the actual output at the current time step.

The output dynamics are written as the sum of the linear expression of (5.3) plus an unknown nonlinear correction term $\Psi(\mathbf{x}_k, \mathbf{u}_k)$

$$\mathbf{y}_{k+1} = \mathbf{C}\Phi(\mathbf{x}_k - \mathbf{x}^o) + \mathbf{C}\Gamma(\mathbf{u}_k - \mathbf{u}^o) + \mathbf{y}^o + \Psi(\mathbf{x}_k, \mathbf{u}_k)$$

The current measurement or estimate of the output can be used to estimate the past value of the correction term

$$\hat{\Psi}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) = \mathbf{y}_k - \mathbf{C}\Phi(\mathbf{x}_{k-1} - \mathbf{x}^o) - \mathbf{C}\Gamma(\mathbf{u}_{k-1} - \mathbf{u}^o) - \mathbf{y}^o$$

Assuming that the variation in the nonlinear correction term is small between control cycles implies that

$$\Psi(\mathbf{x}_k, \mathbf{u}_k) \approx \Psi(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$$

so that the current estimate is given by

$$\hat{\Psi}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{y}_k - \mathbf{C}\Phi(\mathbf{x}_{k-1} - \mathbf{x}^o) - \mathbf{C}\Gamma(\mathbf{u}_{k-1} - \mathbf{u}^o) - \mathbf{y}^o$$

Thus the *a priori* plus adaptive control becomes

$$\mathbf{u}_k = \mathbf{u}^o + (\mathbf{C}\Gamma)^+ [\mathbf{y}_{d,k+1} - \mathbf{y}^o - \mathbf{C}\Phi(\mathbf{x}_k - \mathbf{x}^o) - \hat{\Psi}_k]$$

Because this estimate of $\hat{\Psi}$ is unstructured (the dependencies of $\hat{\Psi}$ on \mathbf{x}_k and \mathbf{u}_k are not directly observable), this form of adaptive augmentation does not accommodate errors in the Γ matrix or nonlinearities in the control.

The estimation process is illustrated in Fig. 5.9. Note that the raw estimates are filtered since noise on the state measurements will propagate directly into $\hat{\Psi}$, which is a weakness of this adaptive scheme.

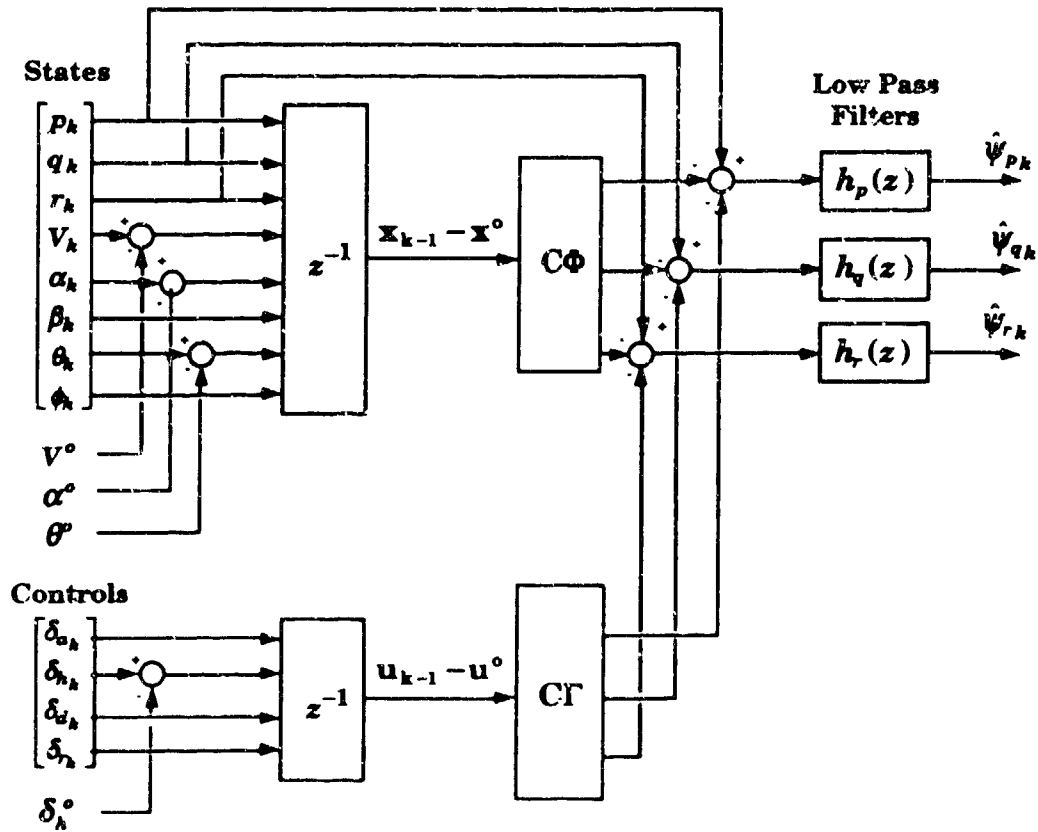


Figure 5.9. Estimation of Nonlinear Correction Terms.

Learning Augmentation

While adaptation is used to compensate for modeling error while the learning system converges, the ultimate goal is to supplant the responsibilities of the adaptive scheme with the learned map as the map becomes accurate. Unlike the adaptive control component, the learning component is less susceptible to noise and is able to compensate for errors in the Γ matrix. This section describes how the learned map is used to generate the learning augmented control signal. The details on how the learned map is generated are discussed in Section 5.6.

The network builds a forward map $\mathbf{n}(\mathbf{x}_k, \mathbf{u}_k)$ for the nonlinear compensation term

$$\mathbf{n}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) = \mathbf{y}_k - \mathbf{C}\Phi(\mathbf{x}_{k-1} - \mathbf{x}^0) - \mathbf{C}\Gamma(\mathbf{u}_{k-1} - \mathbf{u}^0) - \mathbf{y}^0$$

so that the design model for the control is given by

$$\mathbf{y}_{k+1} = \mathbf{C}\Phi(\mathbf{x}_k - \mathbf{x}^0) + \mathbf{C}\Gamma(\mathbf{u}_k - \mathbf{u}^0) + \mathbf{y}^0 + \mathbf{n}(\mathbf{x}_k, \mathbf{u}_k) + \hat{\Psi}_k$$

The adaptive term now accounts for output errors in the new design model (*a priori* plus learned)

$$\hat{\Psi}_k = y_k - C\Phi(\mathbf{x}_{k-1} - \mathbf{x}^o) - C\Gamma(\mathbf{u}_{k-1} - \mathbf{u}^o) - y^o - \mathbf{n}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$$

Since the control objective is $y_{k+1} = y_{d,k+1}$, \mathbf{u}_k must be solved from

$$y_{d,k+1} - C\Phi(\mathbf{x}_k - \mathbf{x}^o) - C\Gamma(\mathbf{u}_k - \mathbf{u}^o) - y^o - \mathbf{n}(\mathbf{x}_k, \mathbf{u}_k) - \hat{\Psi}_k = 0 \quad (5.5)$$

To achieve a closed-form solution for \mathbf{u}_k it is necessary to linearize the network term about the current state and previous control

$$\mathbf{n}(\mathbf{x}_k, \mathbf{u}_k) \approx \mathbf{n}(\mathbf{x}_k, \mathbf{u}_{k-1}) + \left. \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \right|_{\mathbf{x}_k, \mathbf{u}_{k-1}} (\mathbf{u}_k - \mathbf{u}_{k-1}) \quad (5.6)$$

substitute (5.6) into (5.5) and solve for the current control \mathbf{u}_k

$$\mathbf{u}_k = \left(C\Gamma + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \right)^+ \left[y_{d,k+1} - y^o - C\Phi(\mathbf{x}_k - \mathbf{x}^o) - \hat{\Psi}_k + C\Gamma \mathbf{u}^o - \mathbf{n}(\mathbf{x}_k, \mathbf{u}_{k-1}) + \frac{\partial \mathbf{n}}{\partial \mathbf{u}} \mathbf{u}_{k-1} \right]$$

5.5.3 Sideslip Control

The objective of the sideslip controller is to maintain coordinated flight by regulating β to zero. The β controller is the outer loop that feeds values of commanded stability-axis yaw rate r_c to the inner rate loop. Thus, r_c is the "control" signal of the outer loop. Since the outer-loop dynamics are naturally slower than those of the angular rate dynamics, it is reasonable to neglect the inner-loop dynamics when designing the outer loop; i.e., the dynamics from r_c to r are considered to be high frequency "actuator" dynamics for the purposes of the β controller design. This approach is feasible as long as the bandwidth of the β loop (as dictated by the β error dynamics) is sufficiently lower than that of the inner loop.

The structure of the β controller is very similar to the rate control structure, with the exception that there is no need for a reference model since the control objective is regulation. The β controller does have error dynamics and plant inversion components that perform the same functions as those in the inner-loop (see Fig. 5.10). The error dynamics are essentially identical to those of the rate controllers, and include both proportional and integral feedback of the error signal. The plant inversion process is also very similar.

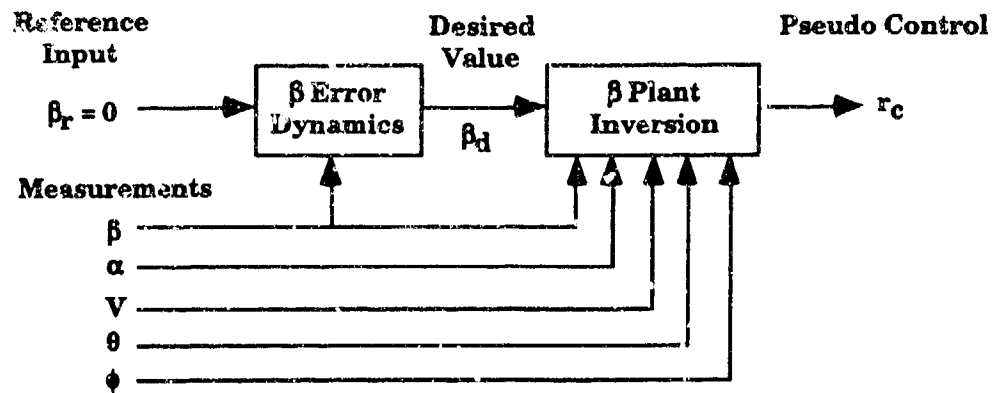


Figure 5.10. The β Regulator.

The expression for the β dynamics is given in body coordinates as [Brumbaugh (1991)]:

$$\begin{aligned} \dot{\beta} = & [D \sin \beta + Y \cos \beta - X_T \cos \alpha \sin \beta + Y_T \cos \beta - Z_T \sin \alpha \sin \beta] / V m \\ & + g [\sin \theta \cos \alpha \sin \beta + \cos \theta \sin \phi \cos \beta - \cos \theta \cos \phi \sin \alpha \sin \beta] / V \\ & + p \sin \alpha - r \cos \alpha \end{aligned} \quad (5.7)$$

The first term on the right-hand side represents the effect of aerodynamic and thrust forces on β . It is a poorly known, complex, nonlinear function of the states. As such, it is very difficult to precompute, and is best accommodated via the adaptive and learning components of the controller. The second and third terms, on the other hand, are relatively simple, well-defined, and easy to compute (assuming that measurements or estimates of the relevant states are available). The last two terms dominate the expression and are equal to the stability-axis yaw rate; i.e., the "pseudo" control of the outer-loop:

$$r_s = r \cos \alpha - p \sin \alpha$$

The details of the *a priori*, adaptive, and learning components of the plant inversion are provided below.

A Priori Component of β Plant Inversion

The β equation may be discretized using a simple Euler approximation

$$\begin{aligned} \beta_{k+1} = & \beta_k + \Delta T g \left[\begin{array}{c} \sin \theta_k \cos \alpha_k \sin \beta_k + \cos \theta_k \sin \phi_k \cos \beta_k \\ - \cos \theta_k \cos \phi_k \sin \alpha_k \sin \beta_k \end{array} \right] / V_k \\ & + \Psi_k - \Delta T r_s \end{aligned} \quad (5.8)$$

where the effect of the aerodynamic and thrust forces are treated as unknown dynamics and are lumped into the Ψ term. The objective of plant inversion is to find the control r_c that achieves the desired value of sideslip at the next control cycle. The *a priori* component of the control is determined by solving this equation for the commanded value of stability-axis yaw rate (and neglecting Ψ),

$$r_{c,s} = -\frac{1}{\Delta T}(\beta_{d,k+1} - \beta_k) + \frac{g}{V_k} \begin{bmatrix} \sin \theta_k \cos \alpha_k \sin \beta_k + \cos \theta_k \sin \phi_k \cos \beta_k \\ -\cos \theta_k \cos \phi_k \sin \alpha_k \sin \beta_k \end{bmatrix} \quad (5.9)$$

Adaptive and Learning Components of β Inversion

Equation (5.8) may be written as

$$y_{k+1} = f(\mathbf{x}_k) + bu_k$$

where $y = \beta$, $u = r_c$, and $f(\mathbf{x}_k) + bu_k$ represents the *a priori* component of the β dynamics (the right-hand side of (5.8)). The network will build a forward map $n(\mathbf{x}_k, n_k)$ for the aerodynamic and thrust forces that are absent from (5.8)

$$n(\mathbf{x}_{k-1}, n_{k-1}) = y_k - f(\mathbf{x}_{k-1}) - bu_{k-1}$$

so that the new design model with learning is

$$y_{k+1} = f(\mathbf{x}_k) + bu_k + n(\mathbf{x}_k, n_k) \quad (5.10)$$

The adaptive system will account for output errors in (5.10)

$$\hat{\Psi}_k = y_k - f(\mathbf{x}_{k-1}) - bu_{k-1} - n(\mathbf{x}_{k-1}, n_{k-1})$$

and the complete design model is given by

$$y_{k+1} = f(\mathbf{x}_k) + bu_k + n(\mathbf{x}_k, n_k) + \hat{\Psi}_k \quad (5.11)$$

The control objective is $y_{k+1} = y_{d,k+1}$, so that the control must be solved from (5.11).

Recall from the inner-loop problem, that the network dynamics had to be linearized to obtain a closed-form solution. A similar procedure must be carried out here, resulting in

$$u_k = \frac{y_{d,k+1} - f(\mathbf{x}_k) - n(\mathbf{x}_k, u_{k-1}) + \frac{\partial n}{\partial u} u_{k-1} - \hat{\Psi}_k}{b + \frac{\partial n}{\partial u}}$$

5.5.4 Learning System

A linear-Gaussian network with an incremental gradient learning algorithm was used to form the basis of the learning system in this example. The input/output equations for this network are repeated below (see Section 3.5 for more detail):

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^n \Gamma_i(\mathbf{x}) \mathbf{f}_i(\mathbf{x})$$

where \mathbf{x} and \mathbf{y} are the input and output vectors, respectively, n is the number of nodes in the network, $\mathbf{f}_i(\mathbf{x})$ are the local basis functions, and $\Gamma_i(\mathbf{x})$ are the normalized influence functions, which are defined to be

$$\Gamma_i(\mathbf{x}) = \frac{\gamma_i(\mathbf{x})}{\sum_{j=1}^n \gamma_j(\mathbf{x})} \quad \text{with} \quad 0 < \Gamma_i(\mathbf{x}) \leq 1 \quad \text{and} \quad \sum_{i=1}^n \Gamma_i(\mathbf{x}) = 1$$

In the case of a linear-Gaussian network, the functions $\mathbf{f}_i(\mathbf{x})$ and $\gamma_i(\mathbf{x})$ become

$$\begin{aligned} \mathbf{f}_i(\mathbf{x}) &= \mathbf{M}_i(\mathbf{x} - \mathbf{x}_i^0) + \mathbf{b}_i \\ \gamma_i(\mathbf{x}) &= c_i \exp\left\{-(\mathbf{x} - \mathbf{x}_i^0)^T \mathbf{Q}_i (\mathbf{x} - \mathbf{x}_i^0)\right\} \end{aligned}$$

For the work presented here, the matrices \mathbf{M}_i and the vectors \mathbf{b}_i were adjustable, but the matrices \mathbf{Q}_i (each \mathbf{Q}_i must be symmetric positive definite), the vectors \mathbf{x}_i^0 , and the scalars c_i were all held constant. As shown in Fig. 5.11, a total of $n = 27$ linear-Gaussian node pairs were used in this network.¹ The network had eight inputs covering Mach number, angle-of-attack, sideslip, dynamic pressure, as well as aileron, horizontal stabilator, differential stabilator, and rudder inputs. The four outputs of the network represent learned (but initially unmodeled) dynamics in roll rate, pitch rate, yaw rate, and sideslip as a function of the eight inputs.

¹ Although Fig. 5.11 seems to indicate otherwise, the linear-Gaussian nodes in the network are really arranged in a single layer (which has been folded to make the figure more compact).

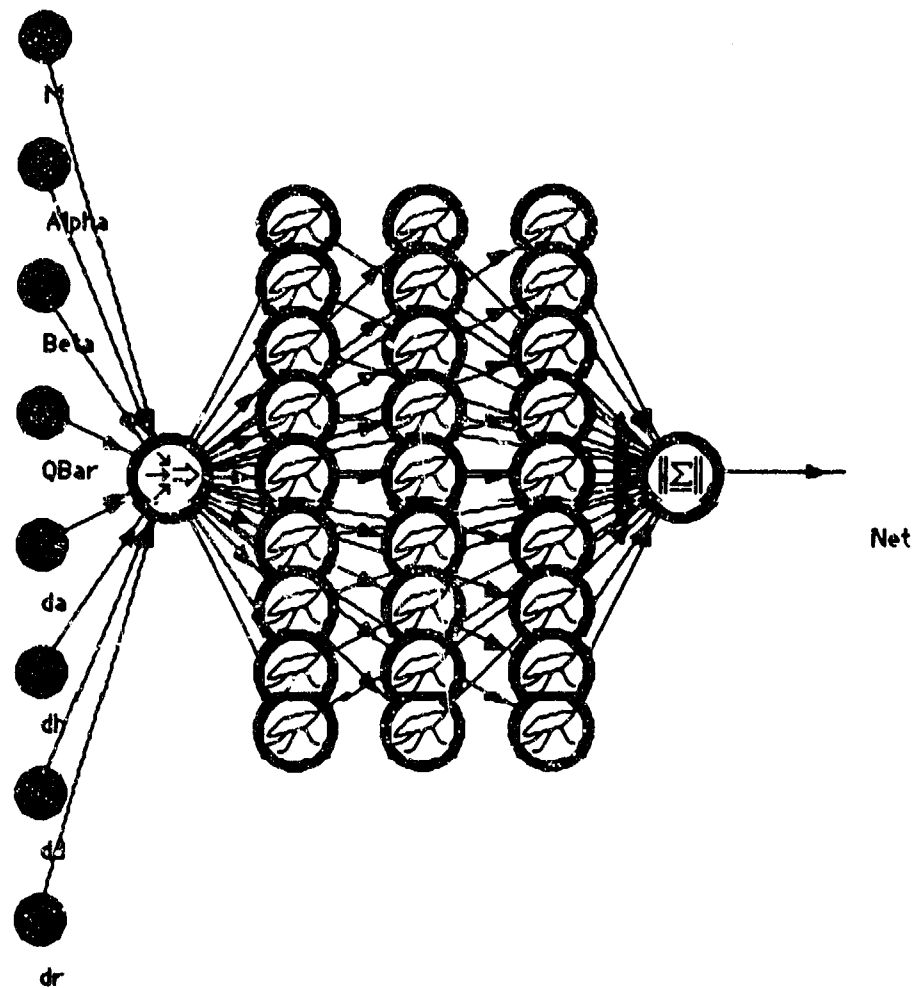


Figure 5.11. Linear-Gaussian Network Used in the Hybrid Controller.¹

¹ This network has eight inputs (Mach number, angle-of-attack, sideslip, dynamic pressure, aileron, horizontal stabilator, differential stabilator, and rudder), 27 nodes arranged in a single layer (but drawn in a more compact form), and four outputs (contained in the vector-valued signal "Net"). The network outputs represent the learned contributions to the prediction of the expected *next* values of aircraft roll rate, pitch rate, yaw rate, and sideslip, in terms of the *current* network inputs.

5.6 Simulation Results

Experimental results obtained from a software simulation of the hybrid adaptive/learning control methodology, applied to the attitude rate control problem on a nonlinear aircraft model and demonstrated relative to the S-trajectory maneuver, are summarized in this section. The basic experimental setup is shown below in Fig. 5.12, which is a snapshot of the *NetSim* project window. The key software modules are listed in Table 5.2.

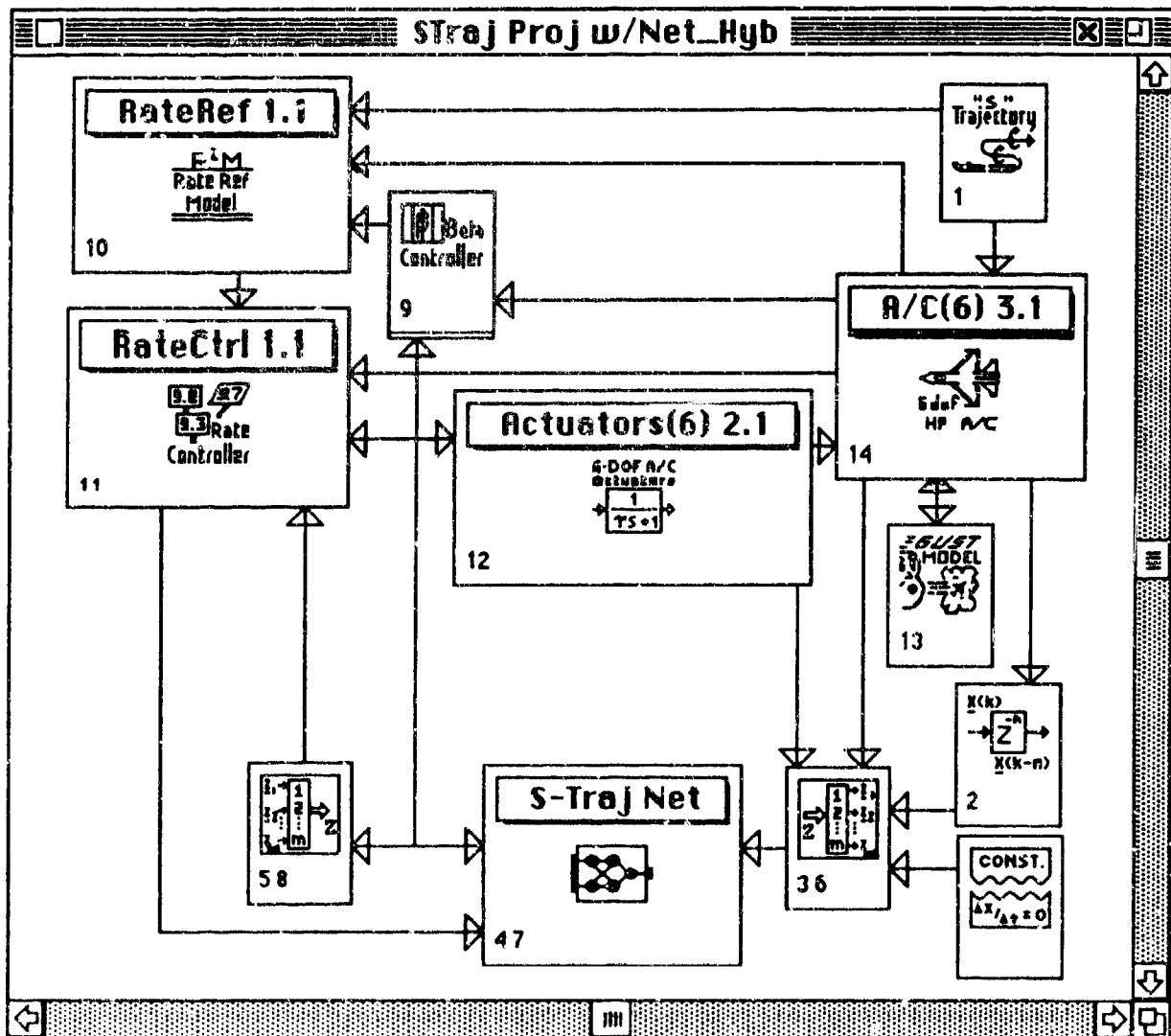


Figure 5.12. A Snapshot of the *NetSim* Project Window Used in the 6-DOF Flight Control Demonstration.

Table 5.2. Main *NetSim* Component Modules for S-Trajectory Demonstration.

| | |
|------------------------|---|
| A/C(6) | 6-DOF, nonlinear aircraft model |
| Actuators(6) | actuator suite for 6-DOF aircraft |
| "S" Trajectory | open-loop guidance command generator |
| RateRef | performance (reference) model w/ error dynamics |
| RateCtrl | attitude rate controller |
| Beta Controller | sideslip controller |
| S-Traj Net | linear-Gaussian network used for learning |
| Gust Model | Dryden model wind gust generator |

5.6.1 S-Trajectory Maneuver

The S-trajectory maneuver used to demonstrate the hybrid adaptive/learning control system in a coupled, multi-axis flight control scenario is outlined below, in Table 5.3. This maneuver is similar to one described in [Stevens & Lewis (1992)]. Starting from a wings-level, trimmed flight condition at an altitude of 5,000 ft and an airspeed of 987 ft/s (Mach 0.9), guidance commands are issued in an open-loop fashion. As outlined in Section 5.5, the overall control augmentation system consists of an outer-loop sideslip regulator and an inner-loop angular rate tracker. Thus, there are two explicit exogenous inputs to the control augmentation system, $p_{s,com}$ and $q_{s,com}$, which are specified by the guidance command generator; additionally, $\beta = 0$ is assumed to be an implicit command.

Table 5.3. S-Trajectory (Open-Loop) Guidance Commands.

| time (s) | guidance command |
|-----------------|---|
| 0 | accelerate forward; hold throttle at full afterburner |
| 5 | initiate pitch pull-up (10 deg/s) |
| 21 | initiate roll right about stability-axis (60 deg/s) |
| 24 | terminate roll right |
| 38 | terminate pull-up |
| 41 | initiate roll left about stability-axis (60 deg/s) |
| 44 | terminate roll left |
| 60 | terminate maneuver |

The S-trajectory is a difficult and challenging maneuver for several reasons (particularly given the capabilities of the specific aircraft model and actuator suite used). The overall maneuver takes the aircraft through a variety of different flight regimes during the course of its execution. A "side view" (altitude vs. ground track) of the maneuver is shown in Fig. 5.13; this perspective clearly illustrates why the maneuver is referred to as the "S-trajectory." Note that in Fig. 5.13, as well as in all subsequent figures relating to the S-trajectory, the results shown are for the hybrid attitude rate control system, with adaptive and learning augmentation, after learning has occurred.

Figs. 5.14-5.18 provide additional perspectives that are useful for characterizing the S-trajectory maneuver: Fig. 5.14 shows the Euler angles associated with the maneuver; Fig. 5.15 is a plot of altitude vs. airspeed; Fig. 5.16 shows angle-of-attack and sideslip as a function of time; Fig. 5.17 shows load factor vs. time; and finally, Fig. 5.18 shows dynamic pressure vs. time.

It should be clear from these plots that the S-trajectory is a complex maneuver. For instance, altitude ranges from 5,000 to 22,000 ft, while airspeed ranges from 370 ft/s to 1070 ft/s. Over this spectrum, the dynamic pressure (which is a strong determinant of the effectiveness of the aerodynamic control surfaces) that the aircraft experiences varies from 80 lbs/ft² to 1160 lbs/ft², with the low point coming near the most difficult point in the maneuver (around $t = 40$ s, when the pitch pull-up is terminated, and the second roll is initiated). During this part of the maneuver, the angle-of-attack plummets from around 20 deg to -6 deg and load factor goes negative. In fact, both the angle-of-attack and load factor are negative during the second roll 180 deg roll.

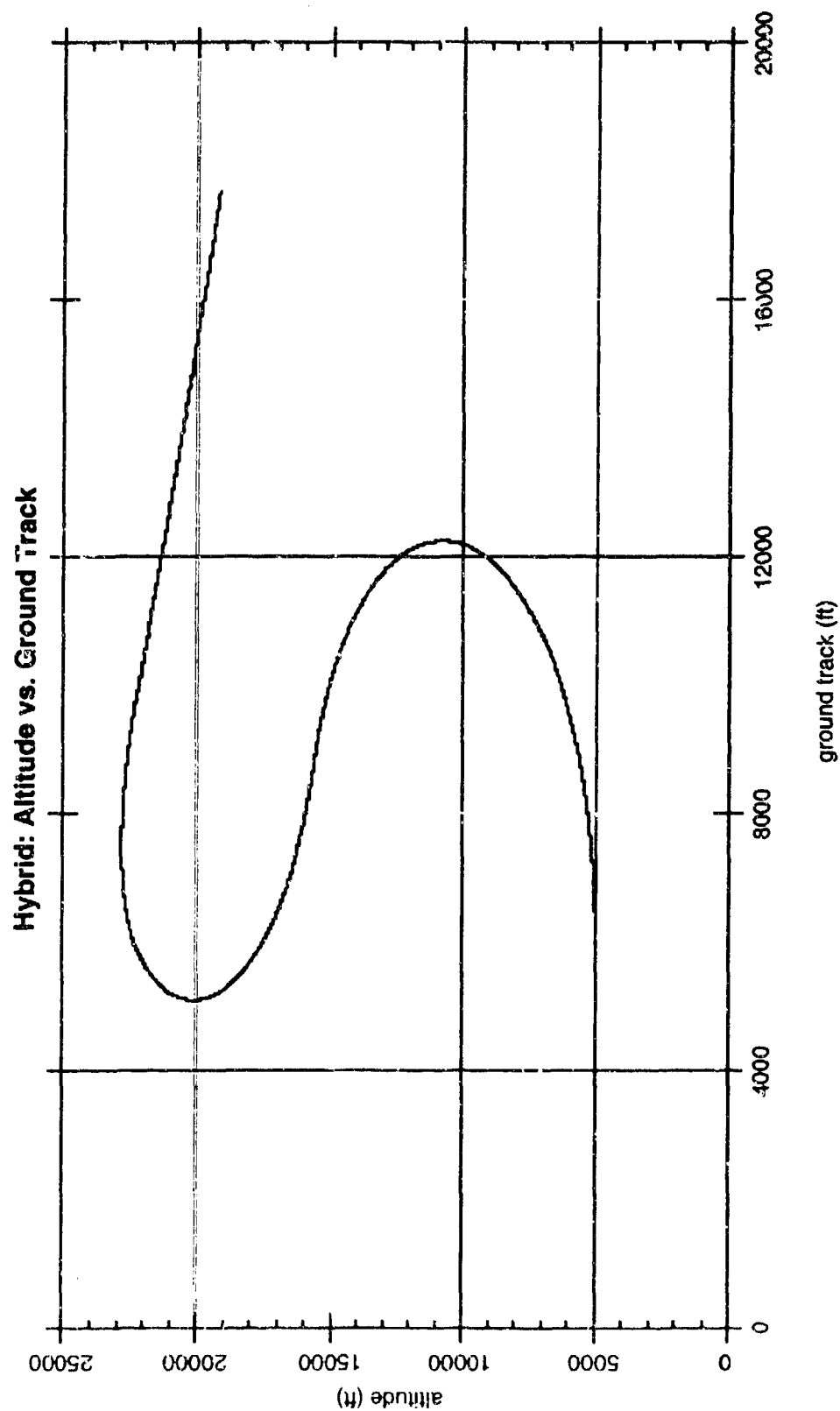


Figure 5.13. A "Side View" of the S-Trajectory Maneuver Using the Hybrid Controller.

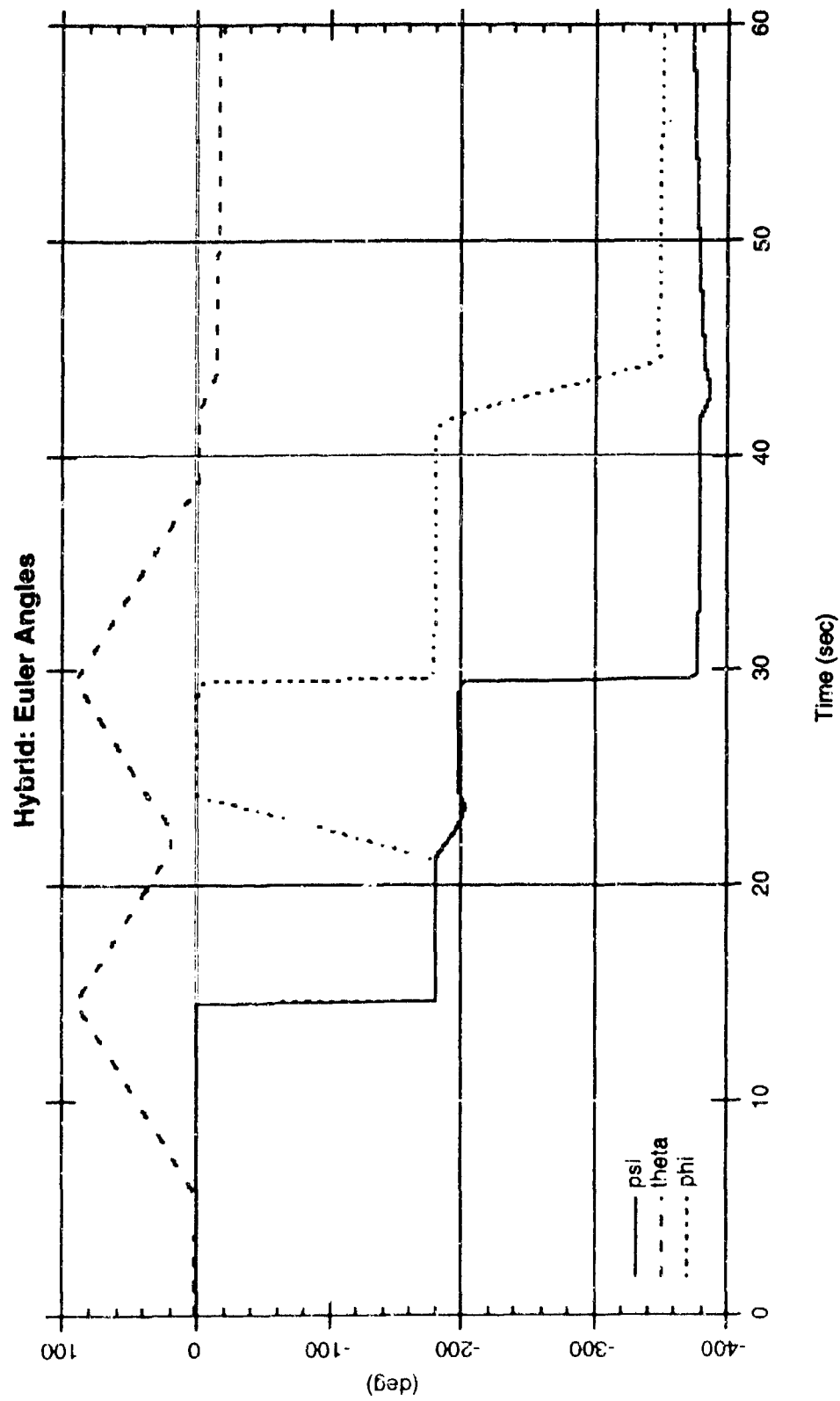


Figure E 14 Euler Angles Associated with the Centimeter Manometer Using the Hybrid Control

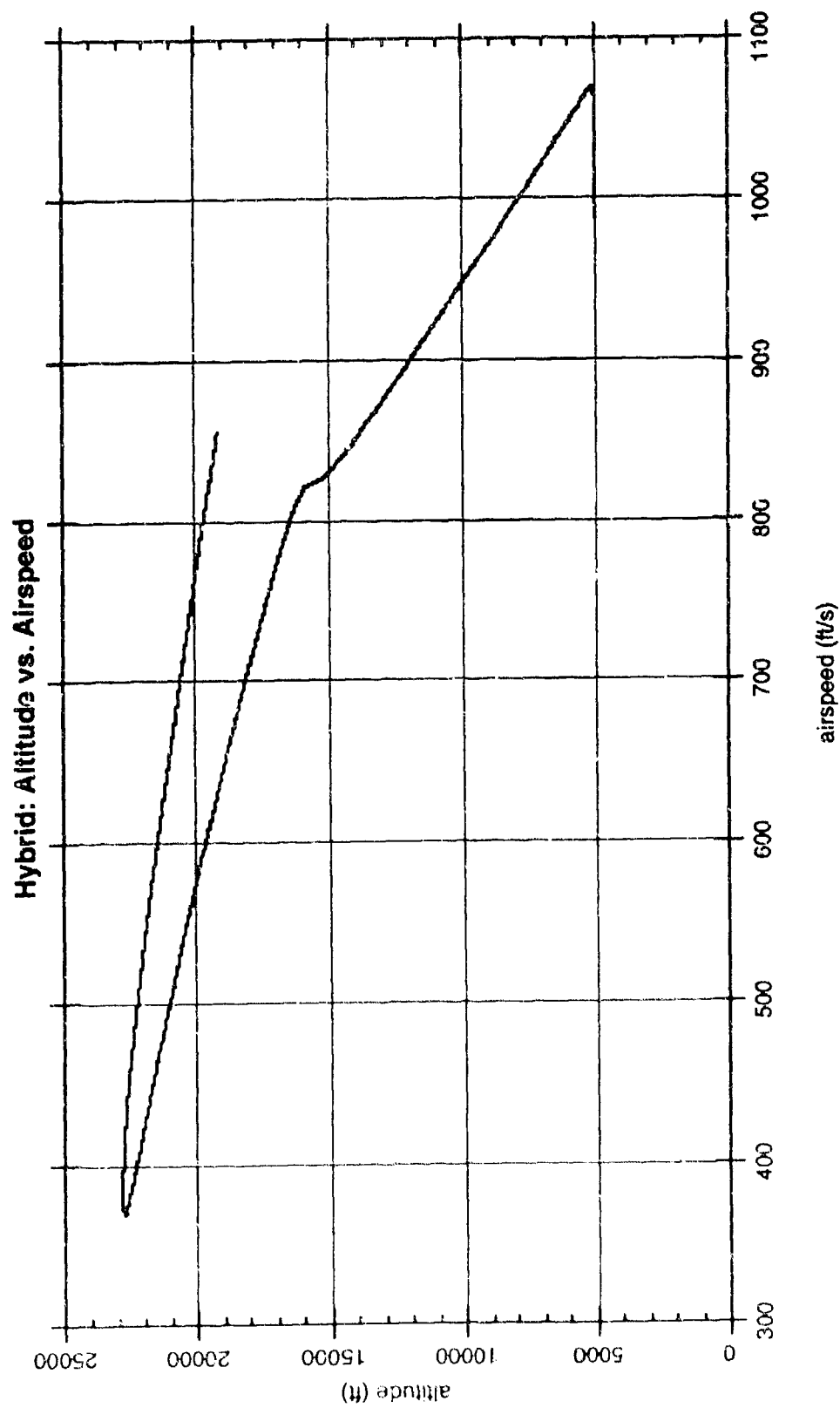
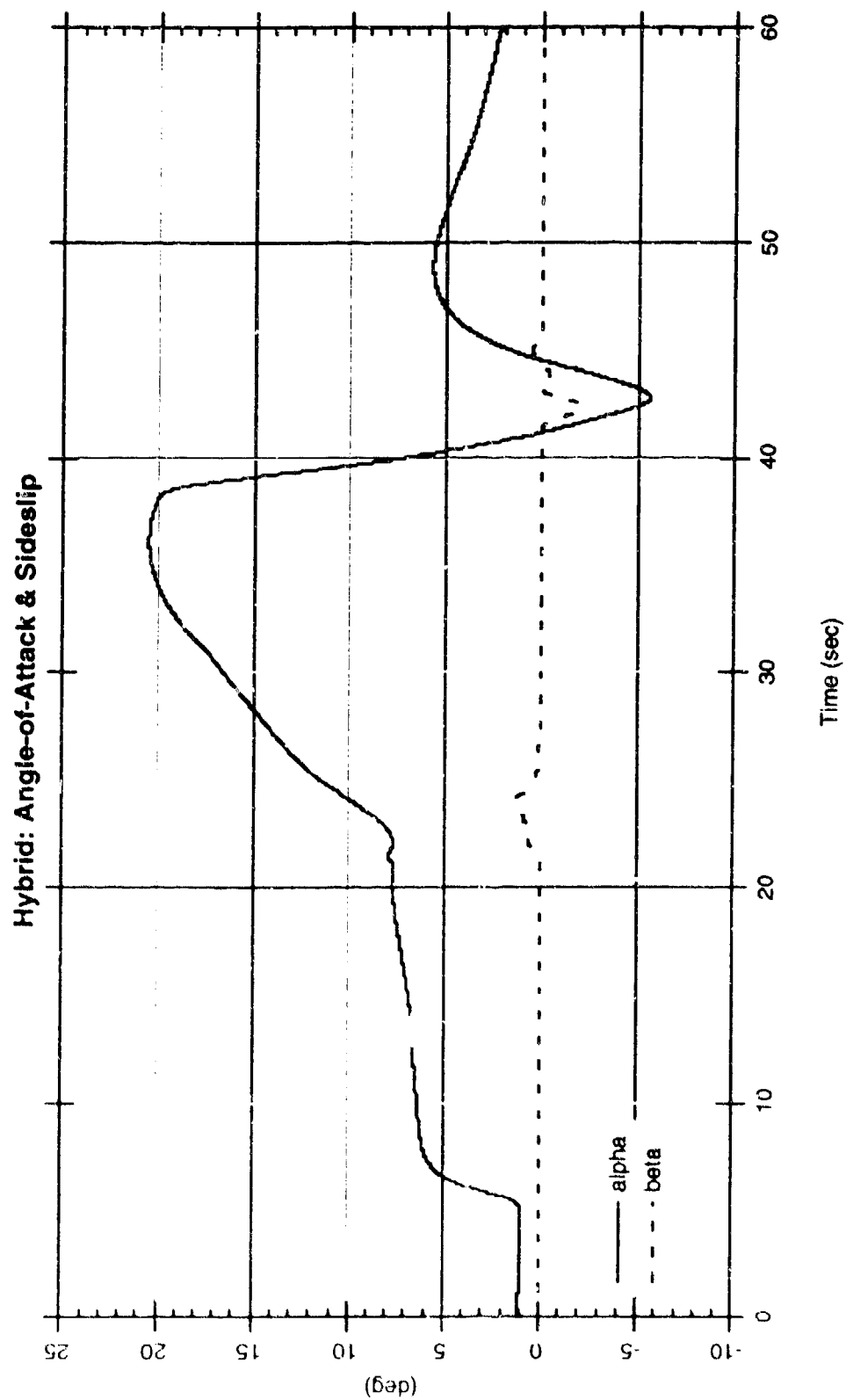


Figure 5.15. S-Trajectory Using the Hybrid Controller: Altitude vs. Airspeed.



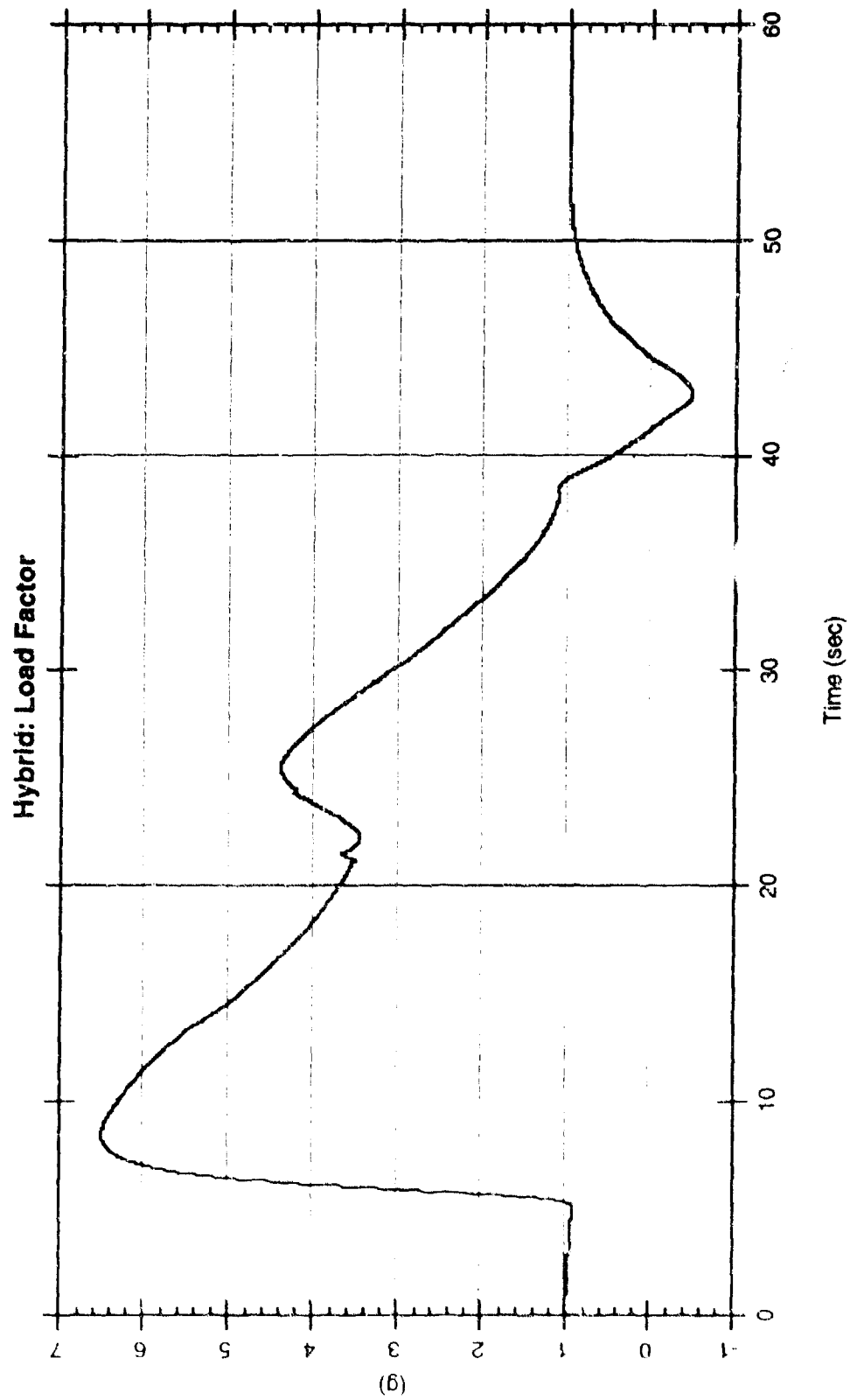


Figure 5.17. S-Trajectory Maneuver Using the Hybrid Control Law: Load Factor vs. Time

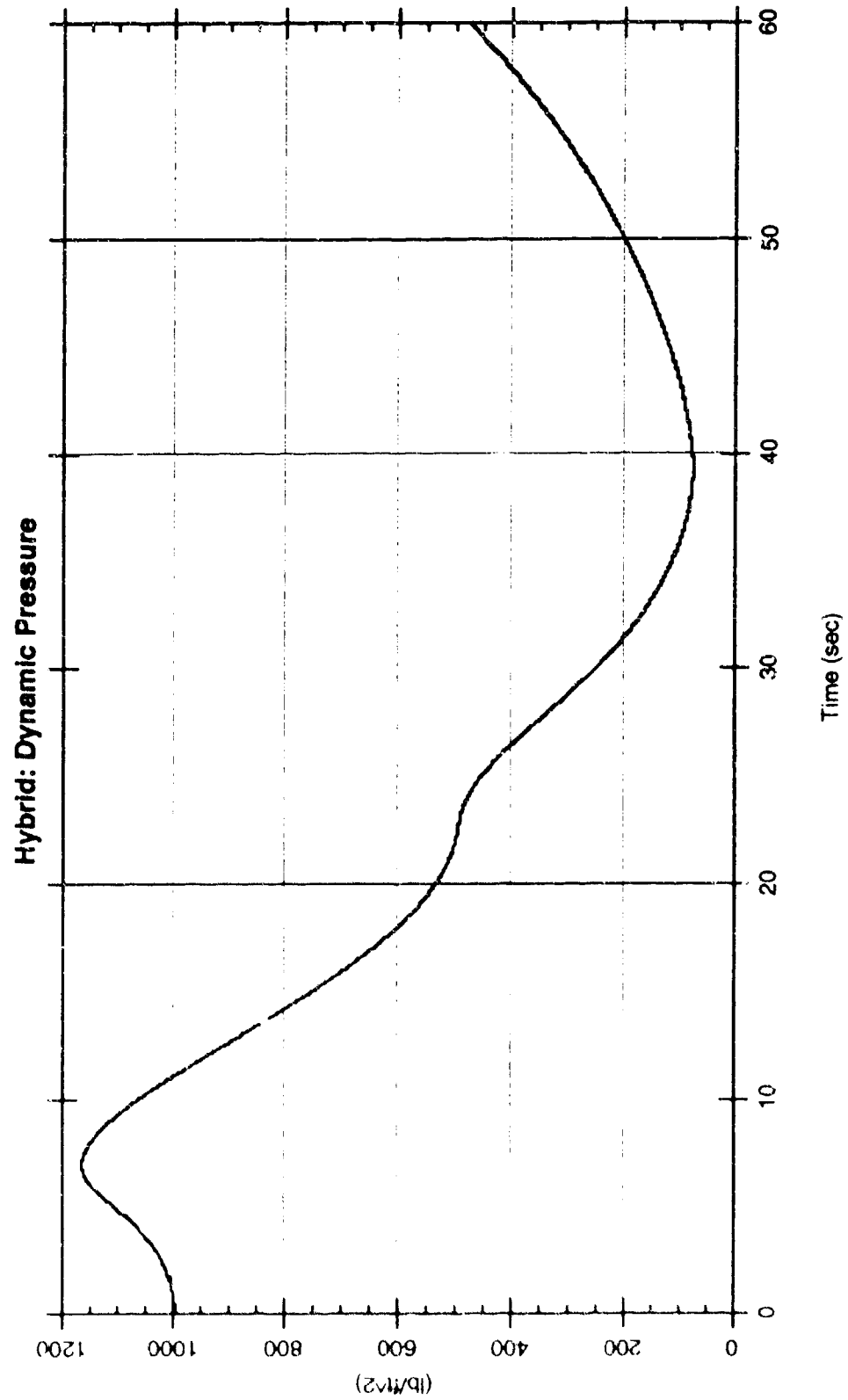


Figure 5.18. S-Trajectory Maneuver Using the Hybrid Controller: Dynamic Pressure vs. Time.

5.6.2 Hybrid Control Tracking Results

The hybrid adaptive/learning control system was trained by exposing it to approximately 4,300 instances of the S-trajectory maneuver. During roughly half of these trials, Dryden model wind gusts [MIL-STD-1797A (1990)] were used to generate disturbances to the otherwise deterministic vehicle dynamics. Although the performance of the learning system in this case (i.e., its ability to *accurately* synthesize the desired unmodeled dynamics in an *efficient* manner, given the *available* resources and experiential data) was adequate for the purposes of this investigation, we believe that much more efficient methods are possible.¹

Initially, before any learning has occurred, the performance of the hybrid control system is identical to that of the control system with adaptive augmentation only. In this case, the adaptive control system acting on its own is able to complete the maneuver, but tracking performance is not very good and, moreover, repeated trials do not make the adaptive controller perform better. With additional experience, the hybrid controller is able to perform better than the adaptive system, due to the incorporation of learning. Note that the *a priori* control system alone (without adaptive nor learning augmentation) is unable to control the vehicle well enough for the maneuver to be completed. In each case, the only *a priori* model information used to design the controllers was a single, low-order linearization of the actual nonlinear aircraft dynamics at a trimmed flight condition corresponding to an altitude of 5,000 ft and an airspeed of 600 ft/s, together with the rigid-body dynamics that appears in (5.8).

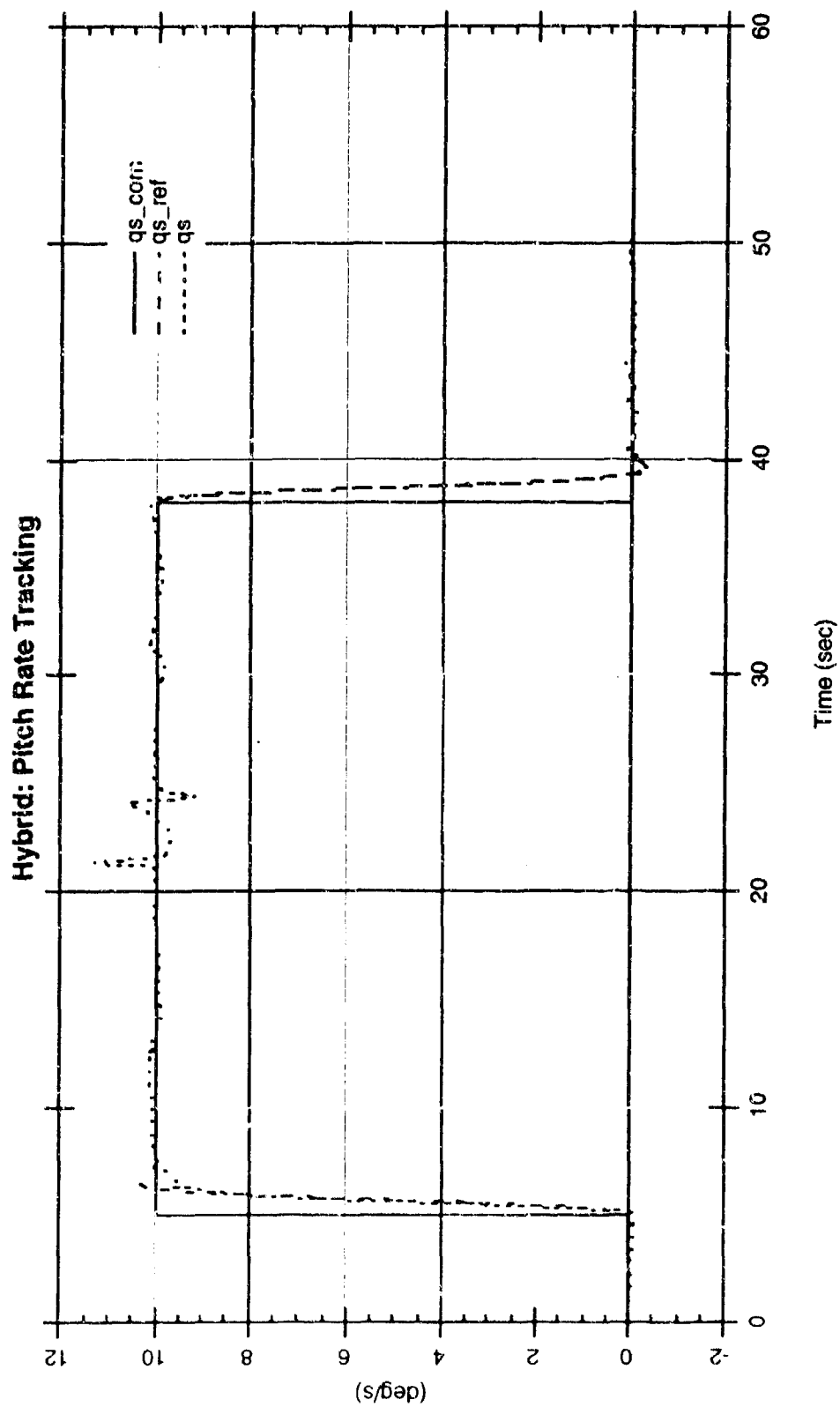
Fig. 5.19 shows the tracking performance of the hybrid controller (after learning has occurred) for the stability-axis pitch rate command. In this figure (as well as the next two), three curves are plotted: a command signal, the corresponding signal output from the reference model dynamics, and the actual response of the nonlinear vehicle under hybrid control. Perfect tracking would result if the actual signal matched that output from the reference model. Fig. 5.20 shows the stability-axis roll rate tracking performance, and Fig. 5.21 shows the stability-axis yaw rate tracking performance.

¹ For example, *variable structure* learning methods could have been employed. Some potential improvements that might be made to the learning system are discussed in Section 6.2.

In Fig. 5.21, the "command" signal is not one of the two exogenous inputs provided by the open-loop command generator, and instead is generated by the outer-loop sideslip controller. Note that the sideslip controller was designed under the assumption that the inner-loop attitude rate tracker was perfect in the sense that it had no error and no lag. Of course, the actual inner-loop controller is not perfect, and so this "design separation" assumption is violated. As a result, the performance of the hybrid controller with respect to yaw rate tracking is not as good as it is for roll and pitch rate tracking.

In point of fact, yaw rate tracking (about the stability-axis) was not an explicit goal of the control augmentation system; instead, it was a means for achieving the explicit goal of sideslip regulation. Thus, the actual tracking performance of the hybrid control system should be judged in terms of Figs. 5.16, 5.19, and 5.20. In Fig. 5.16, the sideslip command should be taken to be identically zero, throughout the course of the maneuver.

It should be clear from Figs. 5.16, 5.19, and 5.20 that the tracking performance of the hybrid adaptive/learning is excellent, especially given the limited *a priori* model information available to it and the difficulty of the S-trajectory maneuver. A direct comparison of the performance of the adaptive and hybrid control systems relative to a near-ideal controller will be presented later in this section.



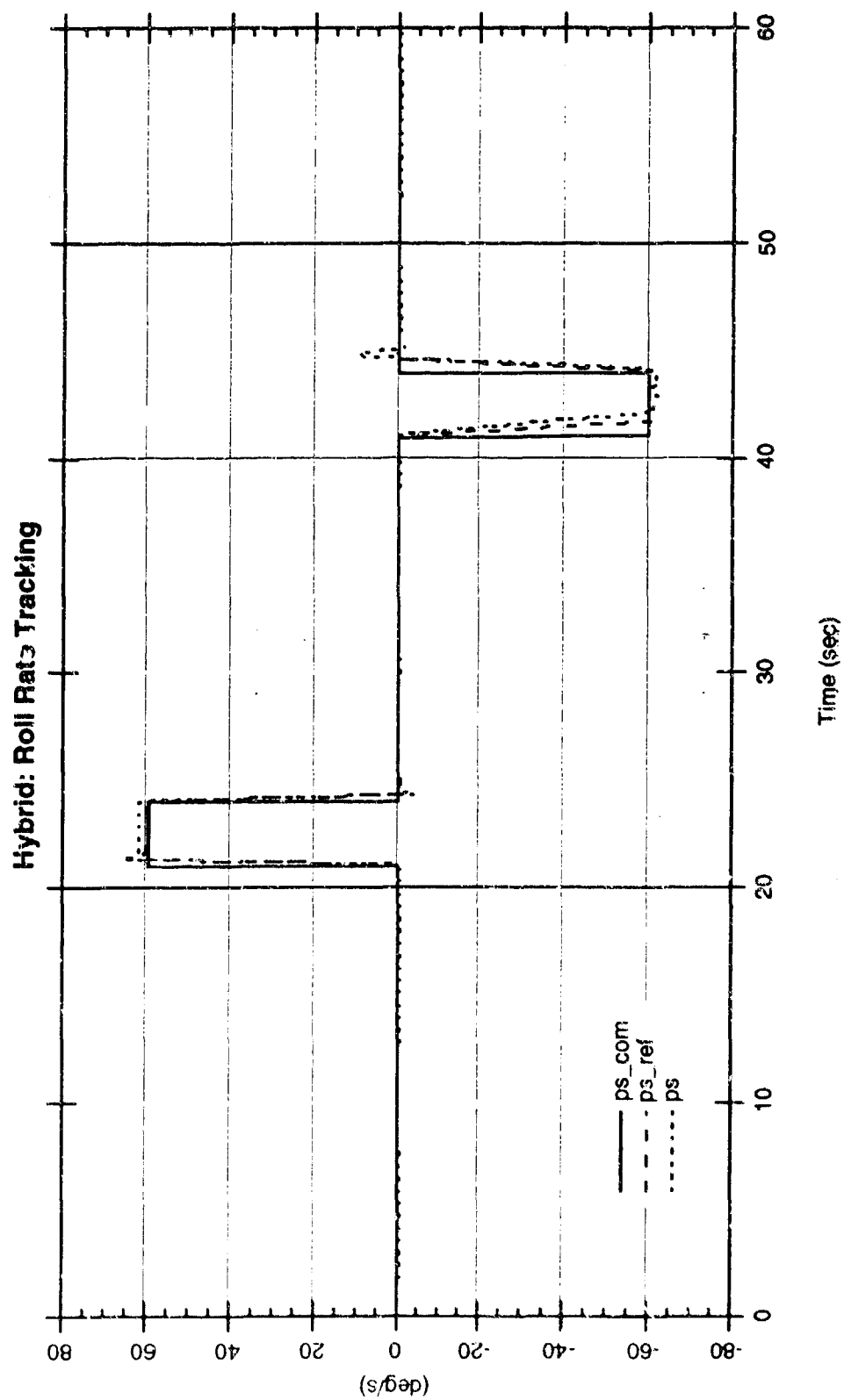


Figure 5.20. S-Trajectory Maneuver Under Hybrid Control: Stability-Axis Roll Rate Tracking

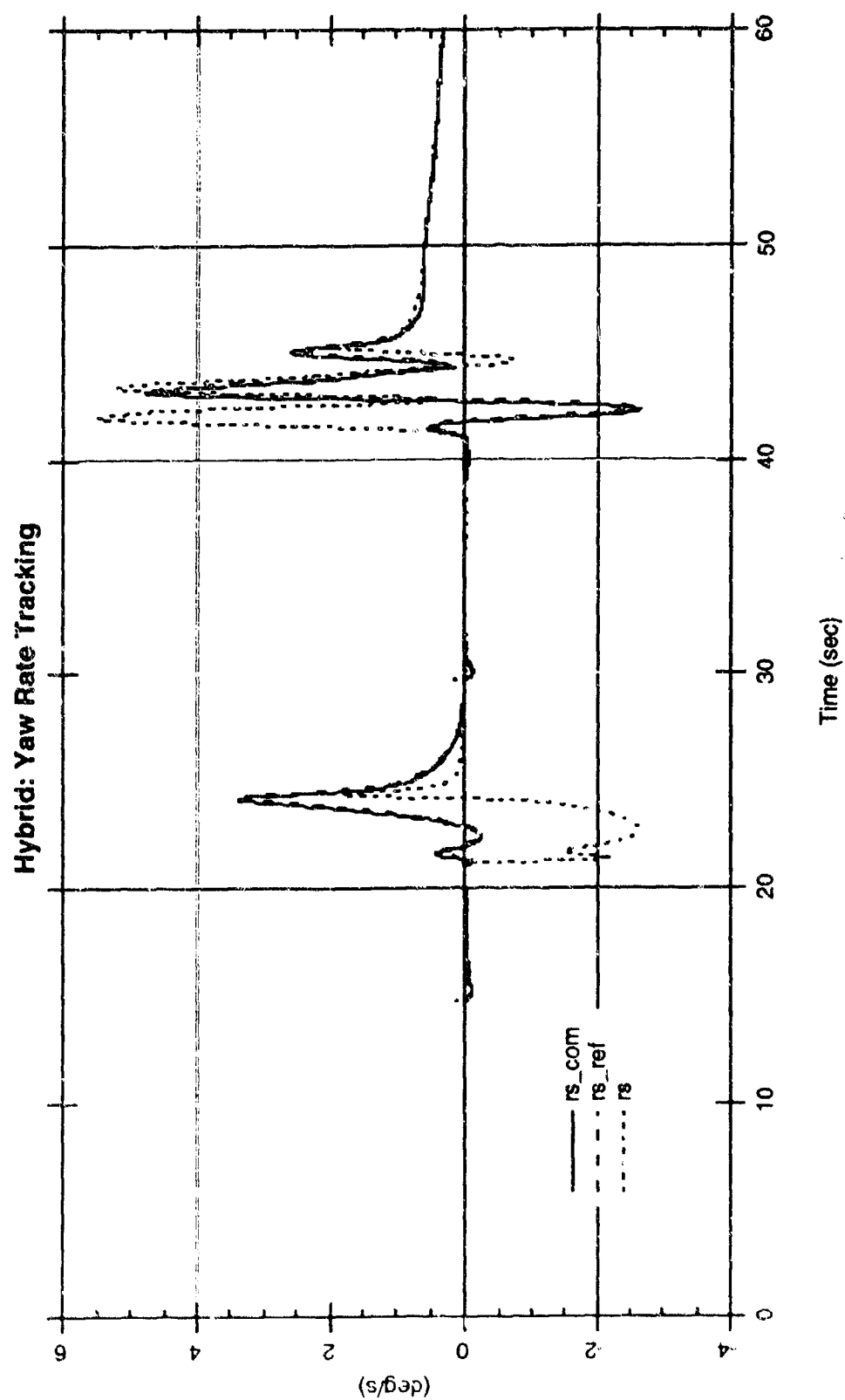


Figure 5.21. S-Trajectory Maneuver Under Hybrid Control: Stability-Axis Yaw Rate Tracking

5.6.3 Hybrid Control Effector Usage

To execute the S-trajectory maneuver, four different aerodynamic control surface effectors were used by the hybrid control system: differential aileron, symmetric horizontal stabilator, differential stabilator, and rudder. Figs. 5.22 through 5.25 show the control effector usage for these surfaces, respectively, during this maneuver.

In each case, the control signals were subject to the position and rate saturation limits of the actuators (as discussed in Section 5.2). As can be readily seen, only the ailerons saturated (around $t = 42$ s, during the second roll). As mentioned previously, the effectiveness of the aerodynamic control surfaces changes radically over the course of the S-trajectory maneuver. In terms of the signals shown in these figures, one can observe this effect by noticing that the magnitude of the control signals required to perform the maneuver is greatest when the dynamic pressure is lowest (roughly from $t = 35$ to 45 s).

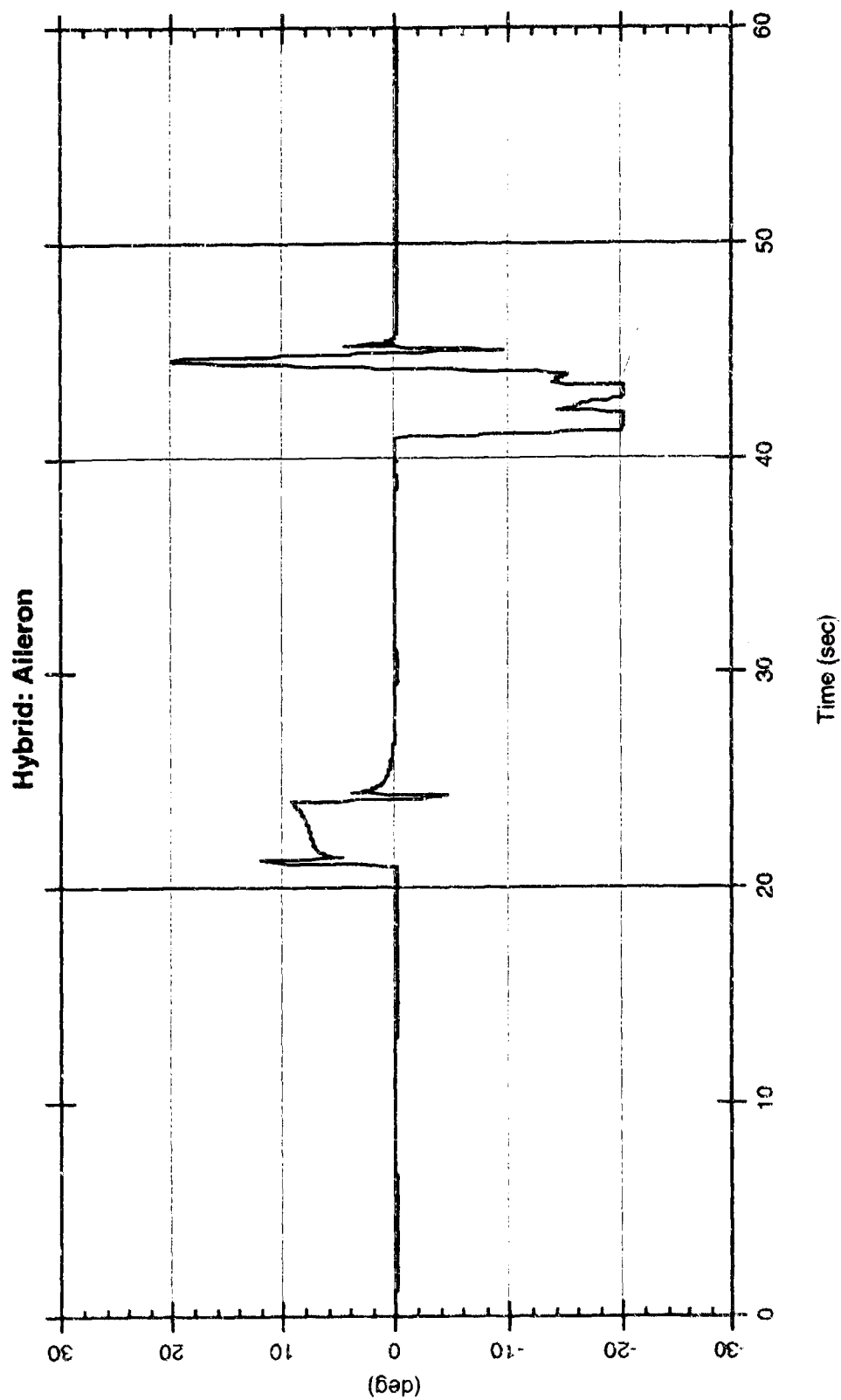


Figure 5.22. S-Trajectory Maneuver Under Hybrid Control: Aileron Response

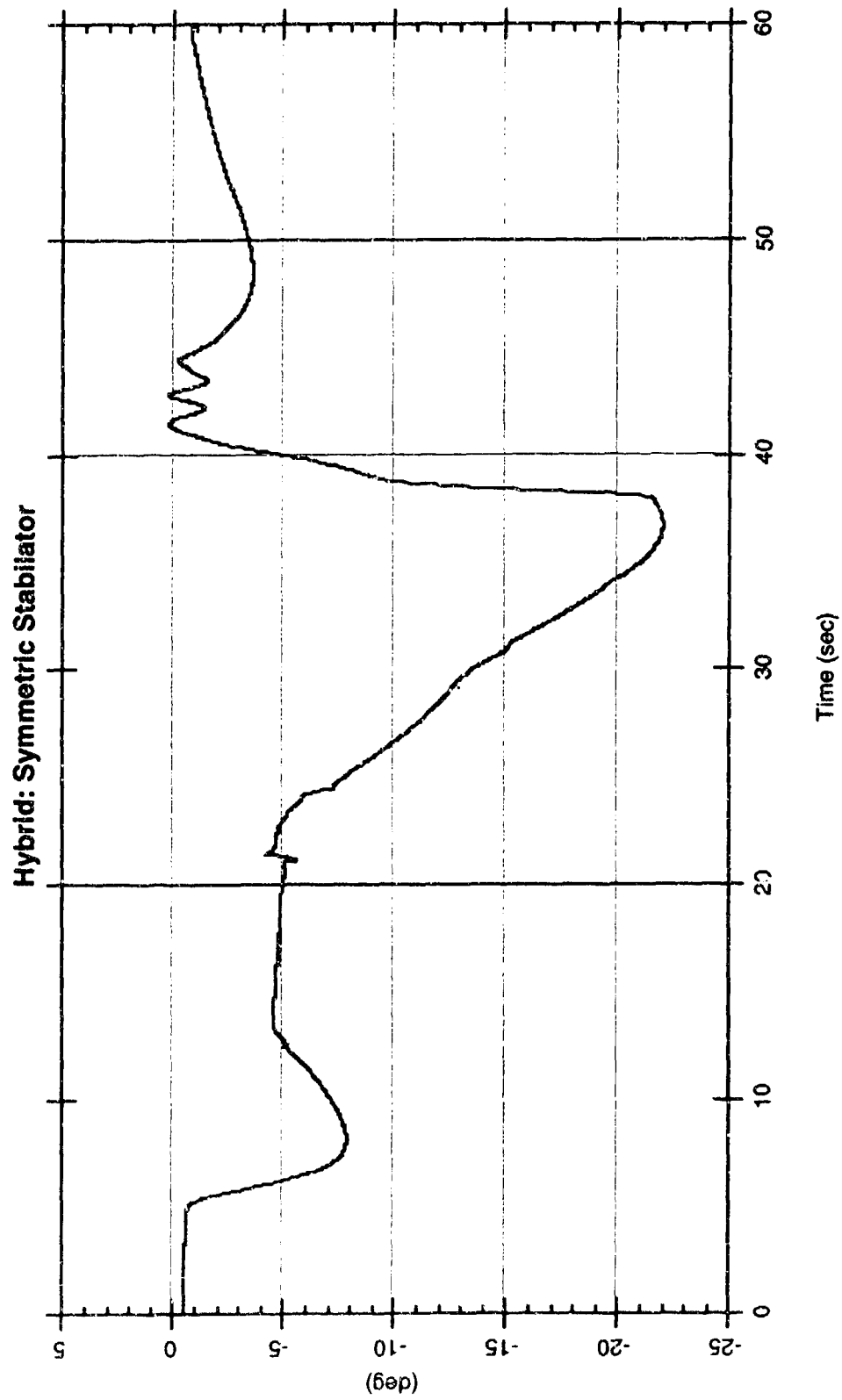


Figure 5.23. S-Trajectory Maneuver Under Hybrid Control: Symmetric Stabilator Response

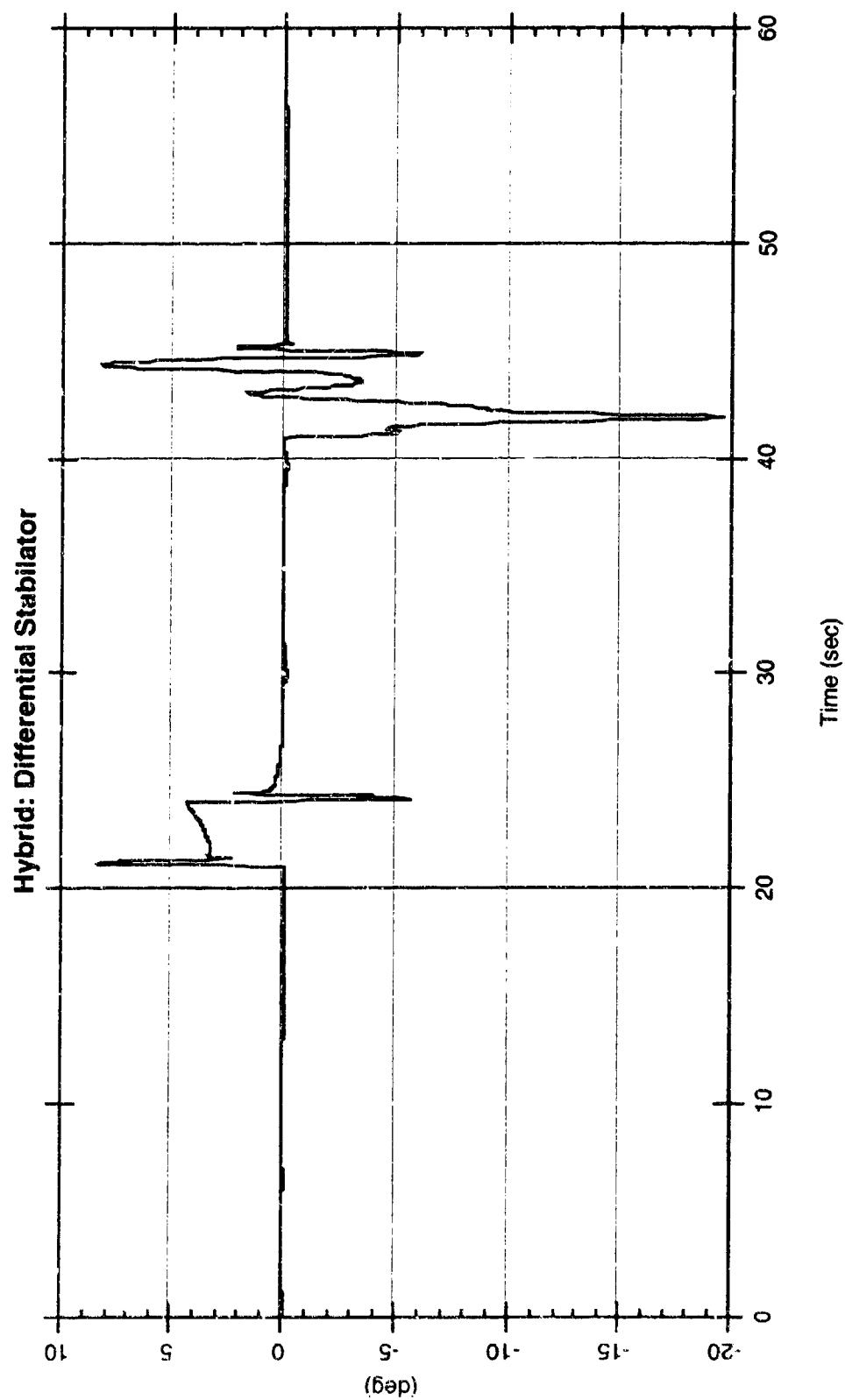


Figure 5.24. S-Trajectory Maneuver Under Hybrid Control: Differential Stabilator Response

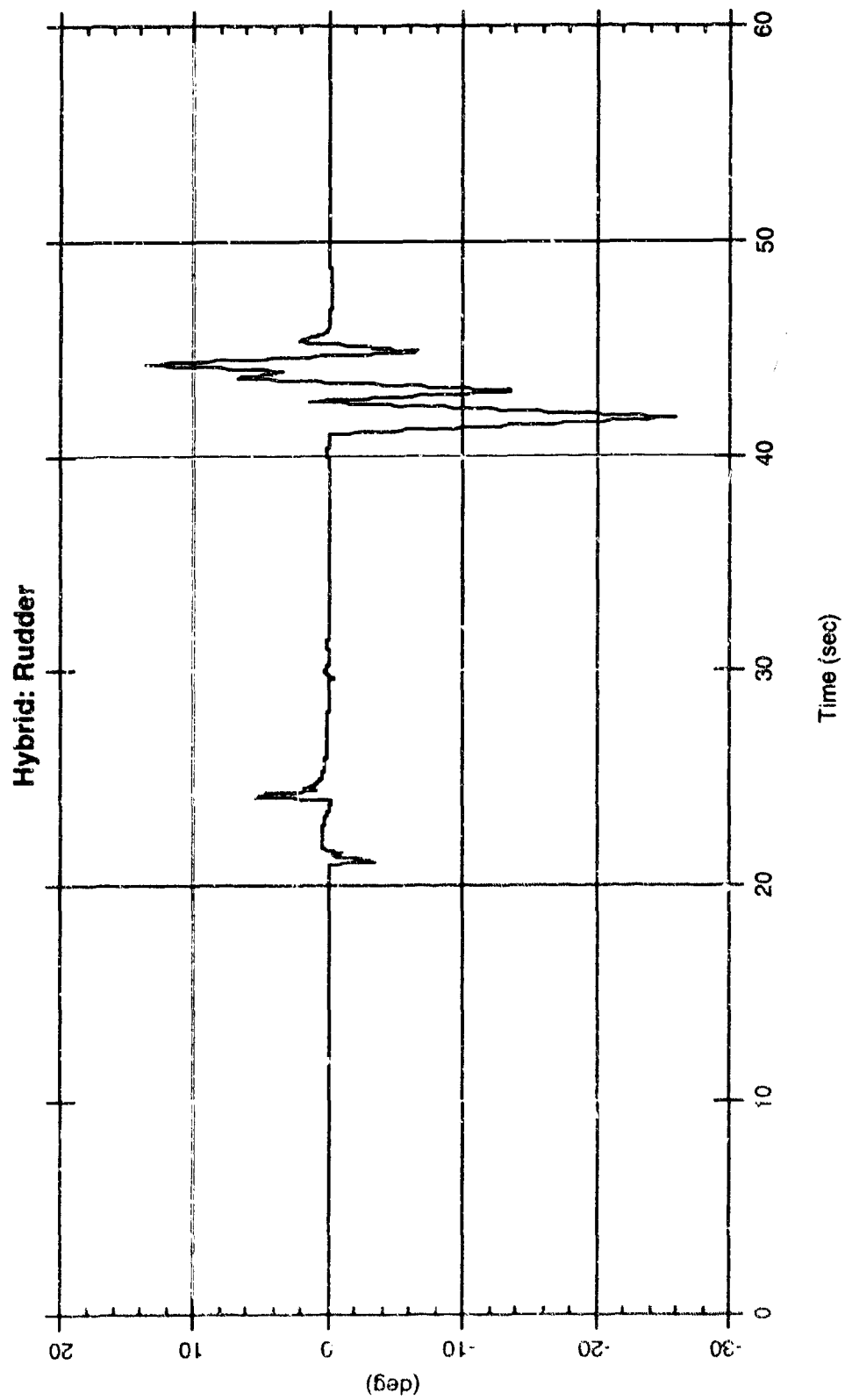


Figure 5.25. S-Trajectory Maneuver Under Hybrid Control: Rudder Response

5.6.4 Summary of Prediction Error Results

The three controllers developed and applied to this problem (linear, adaptive, and hybrid) were all derived using the same *a priori* model information (i.e., a single, constant parameter, low-order linearization of the actual nonlinear aircraft dynamics without actuator or engine dynamics, together with some knowledge of the rigid-body dynamics—not aerodynamics—relating sideslip and yaw rate). A linear compensator was designed using only this *a priori* information. This linear compensator performed so poorly that it could not be used to perform the S-trajectory maneuver (the vehicle would depart from controlled flight).

The adaptive controller was developed as an extension of the linear compensator, by using adaptive augmentation to provide an improved on-line model of the nonlinear aircraft dynamics. A simple adaptive method was incorporated that allowed the vehicle to complete the S-trajectory maneuver, albeit with substantial tracking errors. Similarly, the hybrid controller was developed as an extension to the adaptive controller by using learning augmentation to provide an even better on-line model of the actual nonlinear aircraft dynamics. Prior to learning, the hybrid controller performed identically to the adaptive controller; after a small period of training, the hybrid system was able to perform exceptionally well.

The only real difference between these three controllers was in their ability to identify and predict the unmodeled dynamics of the aircraft. In particular, all three employed the same overall control system architecture, and the same on-line control selection scheme. The linear compensator was of a fixed design, and could not improve its model on-line. Both the adaptive and learning augmented controllers were able to update their models on-line. Thus, the most direct way to compare the performance of the controllers is to examine their ability to *predict* the behavior of the actual nonlinear aircraft in terms of the four outputs of interest: roll rate, pitch rate, yaw rate, and sideslip. Since the linear compensator was unable to execute the maneuver, it was excluded from the comparison. In addition, so as to gauge the relative performance that might be obtained under conditions of near perfect learning, an "ideal" hybrid controller was constructed by replacing the learning system network with modules derived from the actual nonlinear aircraft dynamics. Subsequently, the root-mean-square (RMS) value of the prediction errors for the four outputs were computed over S-trajectory. These re-

sults are summarized in the bar chart shown in Fig. 5.26. Note that the prediction errors in the case of "ideal" augmentation are not quite zero due to the presence of some effects (e.g., actuator position and rate saturation) which could not easily be accounted for and were hence ignored.

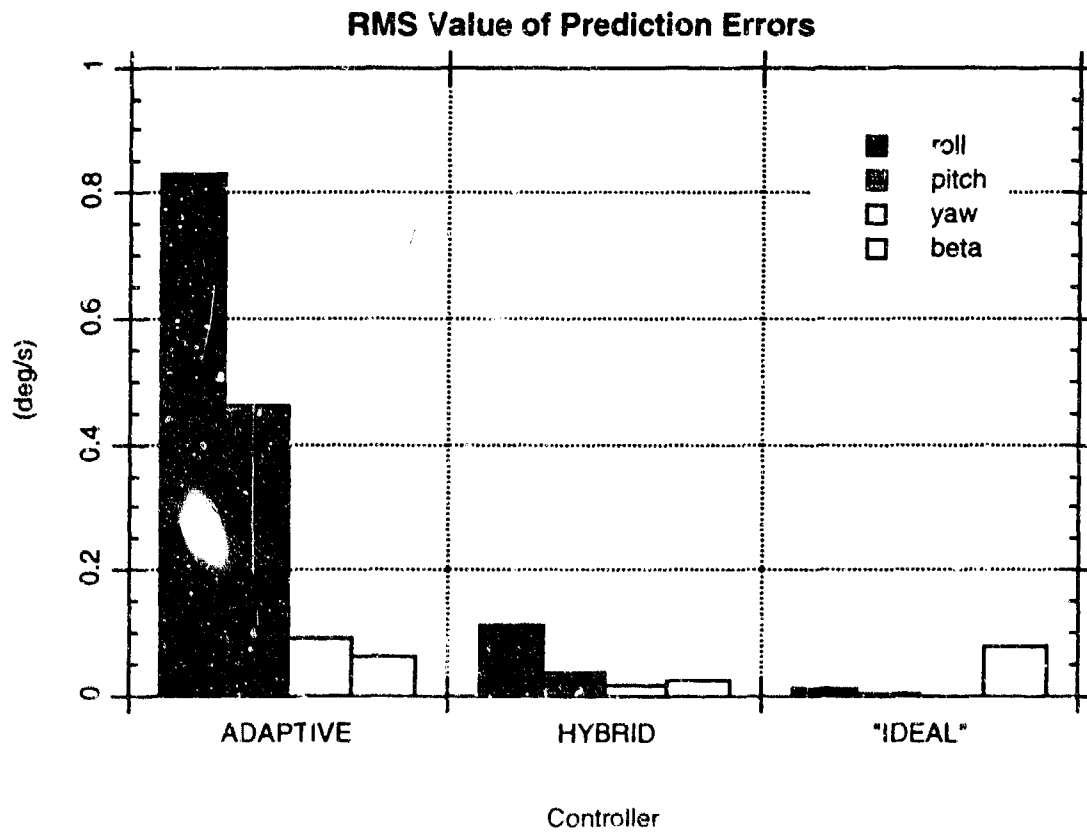


Figure 5.26. Summary of Prediction Errors Using Adaptive, Hybrid Adaptive/Learning, and "Ideal" Augmentation.

Fig. 5.26 clearly shows the improvement that is possible through the use of learning augmentation. The hybrid controller easily outperforms the adaptive system, and is even able to outperform the "ideal" case relative to the sideslip dynamics.

6 Conclusion

Results obtained during this research program clearly demonstrate many of the potential benefits of learning augmented control. At the same time, however, issues uncovered by this investigation also suggest that further work is needed. A summary of the program is provided below in Section 6.1, while topics for future research and development are discussed in Section 6.2. Note that each attachment also includes its own separate set of conclusions and recommendations for future research.

6.1 Summary

This multiphase research program had the broad aim of investigating the application of learning systems to automatic control in general, and to flight control in particular. The first phase analyzed the original drive-reinforcement learning paradigm and examined its application to automatic control, with mixed results. It was shown that while the original algorithm showed promise, it nevertheless lacked the ability to function (alone) as a learning controller. The second phase compared a number of alternative control strategies including conventional linear control, adaptive control, as well as other reinforcement learning control methods (e.g., those developed by Barto, Sutton, et al.). No candidate was found to dominate the field, and none was perceived to be suitable for application to flight control. During this same period, a new hybrid adaptive/learning control scheme was conceived. Subsequently, in the third phase, the hybrid control approach was more fully developed and applied to several nonlinear dynamical systems, including a cart-pole system, aeroelastic oscillator, and a three-degree-of-freedom high performance aircraft. Each application was successful. The fourth phase revisited drive-reinforcement learning from the point of view of optimal control and successfully applied a version embedded in the associative control process architecture to regulate an aeroelastic oscillator. Analysis of this and other similar reinforcement learning approaches indicated that they are best suited to problems in optimal control. The fifth phase examined the problem of learning augmented estimation, and resulted in the development of a preliminary estimation scheme that is consistent with the hybrid adaptive/learning control approach. In the sixth and final phase, the hybrid control methodology was applied to a nonlinear,

six-degree-of-freedom flight control problem, and then successfully demonstrated via a challenging, multiaxis "S-trajectory," maneuver.

Throughout this work, a key concept underlying our approach is the view that "learning" can be interpreted as the automatic synthesis of multivariable functional mappings, based on experiential information that is gained incrementally over time. When combined with adaptation, the resulting hybrid control strategy provides a sophisticated control system design and implementation technique. Moreover, our work emphasizes real-time (on-line) adaptation and learning, and considers the overall problem to have four fundamental elements: adaptation, learning, control, and estimation. For flight control applications, advanced control systems incorporating learning might be used advantageously to:

- *facilitate the control system design and tuning process*
- *accommodate initially unmodeled dynamics*
- *improve performance through on-line self-optimization*
- *improve control usage and efficiency relative to purely adaptive approaches*

The bottom line is that learning augmentation is beneficial to automatic control in general and to flight control, in particular.

In conclusion, the main accomplishments of this program include:

- analysis of the D-R learning paradigm and ACP network architecture in the context of control and optimal control, respectively
- motivation for and identification of issues underlying the application of learning to flight control
- conception and development of the hybrid adaptive/learning control and estimation methodology
- application of a variety of learning systems to the control of a many different nonlinear dynamical systems (e.g., cart-pole system on split-level track, aeroelastic oscillator, and 3-DOF high performance aircraft)
- development of a 6-DOF learning augmented flight control system and demonstration via a multi-axis maneuver
- guidance, supervision, and support of 3 graduate student theses
- production of 11 technical publications

6.2 Recommendations for Future Work

Although significant progress was made during this research program, it is clear that further work is needed. Key topics for future research and development are summarized below.

6.2.1 Reinforcement Learning Systems

The current state of scientific and engineering advancement is such that most reinforcement learning methodologies cannot readily be applied (in a practical sense) to those complex control problems which might warrant such approaches. Even so, the potential benefits associated with a practical reinforcement learning system implementation are significant and not easily overlooked. Moreover, many researchers believe firmly in the existence of such implementations. Thus, we suggest two topics for future research and development in this area.

Reinforcement Learning Applications

A closer examination of the many connections between reinforcement learning and "classical" approaches to solving optimal control and multiplayer game problems is needed. The formulation of many such problems fits the basic scenario of reinforcement learning, with the proviso that reinforcement learning methods are only appropriate for problems in which there is a significant level of uncertainty regarding the plant (or player) dynamics.¹ It is perhaps also true that those who have been approaching such problems from a classical engineering point of view can benefit from the perspective of more biologically motivated researchers, and vice versa.

Continuous Input/Output Reinforcement Learning Systems

One severe limitation of many current reinforcement learning methods is the need to discretize both the problem state space and the control action space. Often,

¹ For example, one might attempt to design *robust* control systems (off-line) using a two-player game scenario, in which one player (the protagonist) attempts to minimize some cost function by selecting an appropriate control law, while the other player (the antagonist) attempts to maximize the same cost function by modifying plant and/or environmental parameters.

only a binary (i.e., bang-bang) control action set is used. Moreover, it appears that there may be great difficulty in scaling such implementations up to the point where a quantized system could be effectively used. An important first step towards making reinforcement learning methods more useful to control theorists and engineers would be to develop systems capable of continuous input/output spaces (i.e., continuous state and action spaces). In addition, variable structure learning (see below) is a feature that almost certainly should be incorporated into reinforcement learning systems.

6.2.2 Learning Systems for Hybrid Adaptive/Learning Control

There are several opportunities for enhancing both the learning system and training process in a hybrid adaptive/learning control system. Generally, these upgrades are aimed at: (i) making the learning process more *efficient*, as well as (ii) automating the learning system design parameter selection process—which is currently done manually.

Variable Learning Rates

When on-line learning is used in control applications, the system state may remain in particular regions of its state-space for extended periods of time during training. Under these conditions, the approximation error should not be expected to tend *uniformly* to zero over the input-space. Instead, the error will be lowest in those areas where the greatest amount of experience has been obtained. This condition leads to conflicting constraints on the learning rate: it should be small (to filter the effects of noise) in those regions where the approximation error is small, but at the same time, it should be large (for fast learning) in those regions where the approximation error is large (relative to the ambient noise level). Resolution of this conflict is possible through the use of *spatially localized learning rates*, where individual learning rate coefficients are maintained for each (spatially localized) region and updated in response to the local learning conditions. Some preliminary work has been performed in this area, e.g., [Jacobs (1988); Berger (1992)].

Variable Structure Algorithms

One potential criticism of the use of learning systems in control applications is that learning may proceed too slowly, so that too much training time is required before the benefits can be realized. To a large extent, slow learning rates are an inherent attribute of *large distributed* networks, since much experiential information is needed to determine the appropriate values for the large number of adjustable parameters. One fundamental approach for achieving more rapid learning is to begin with a small network, having a few adjustable parameters, and incrementally "grow" the network by enlarging its structure. Although such a network initially lacks high representational power, it will be able to quickly capture the main features of the desired mapping. Additional parameters (structure) may then be added to gradually improve the precision of the mapping. Given appropriate logic for adding nodes, this approach should display a high overall learning rate, as the learning process will proceed in a more efficient manner than when using standard gradient-based training algorithms with fixed network structures. Until recently, such "variable structure" algorithms were generally incompatible with the incremental training requirements of on-line control system applications. These issues have recently been addressed to some extent in [Cerrato (1993)].

6.2.3 Learning for Flight Control

Four potential applications of learning to flight control are briefly outlined in this section. It is perceived that the use of learning in these instances might prove to be advantageous, particularly if a learning system were also employed to fulfill the main learning augmented control function discussed in previous chapters.

Learning Augmented Estimation

Successful state estimation typically requires an accurate model of the system. Obviously, learning can be used to facilitate the estimation process if this process is allowed to utilize the modeling information provided by a learning system. The use of learning to provide additional model information should prove to be more robust to measurement noise than if adaptation alone was used. Once learning has occurred, network evaluations provide stored, time-averaged (and more noise

robust) modeling information, whereas adaptation (dealing only with recent temporal sequences) can only gain robustness to noise by filtering these signals (which inherently introduces lag into the estimation process). The material presented in Section 4.3 only represents a first step in this direction—there is much room for further analysis and development.

Learning an Inverse Model

Recall that accurate output tracking in the basic hybrid control approach requires solving a system of algebraic equations for the control variables to obtain the desired output at the subsequent time-step. When the plant model is linear, this solution involves inverting a *constant* matrix that represents input/output control effectiveness. However, in the general nonlinear case, the tracking problem requires solving a nonlinear system of algebraic equations (as the controls no longer enter linearly). In fact, this nonlinear problem may have *many* solutions, or *no* solution at all, depending on the nonlinearities involved. Once a nonlinear tracking equation had been solved at a given flight condition, it might be desirable to *retain* this solution. Generation of the solution could then be expedited in the future if a flight condition in the same vicinity were encountered. In fact, it would be inefficient to numerically solve the same nonlinear problem twice—especially if multiple iterations were required. The generalization implied by these ideas becomes tantamount to *learning an inverse mapping*. Such an approach is generally possible if and only if the solution to the nonlinear tracking problem is unique.

Learning the Trim Manifold

Inaccurate knowledge of the trim manifold can be interpreted as a form of model error. Adding integrators increases controller robustness to such modeling error, but inherently results in slower rates of convergence (since a finite time is required for the integral states to have an impact on the control signal). By using learning to provide the autopilot with a more accurate trim description (in a feed-forward sense), less integral compensation may be required, thereby improving the convergence rate and ultimately resulting in a better autopilot design. Similarly, the state estimation process can benefit from a more accurate characterization of the trim manifold.

Optimizing the Reference System

In model reference control architectures, the controller seeks optimal *tracking performance*. Note that even if the control law perfectly achieves this tracking objective, one can at best only expect to *match* the performance of the chosen reference system—thus, the reference model effectively represents an "upper-bound" on the closed-loop system performance. In such a control paradigm, *global system performance* must be addressed through the design of the reference system. Generally, one would like to construct the best possible reference system that is consistent with the *actual* control capabilities of the system. Furnishing an over-ambitious reference model might introduce instability, whereas selecting a conservative reference system will result in suboptimal closed-loop performance. For general nonlinear systems, performance will be regime dependent—the system may be *capable* of better performance in some regions of the operating envelope than in others. Typically, such variations are not accurately known *a priori*, so that a conservative design may result. Through on-line experience with the actual system, the controller can *learn* to identify troublesome operating regimes and relax the reference model expectations accordingly. Likewise, the reference response can be made more ambitious in those regimes where it is appropriate to do so.

Bibliography

- Albus, J. (1975). "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 97, pp. 220-227.
- Alexander, J., Baird, L., Baker, W., & Farrell, J. (1991). "A Design & Simulation Tool for Connectionist Learning Control Systems: Application to Autonomous Underwater Vehicles," proceedings, *1991 SCS Summer Computer Simulation Conference*.
- Anderson, C. (1989). "Learning to Control an Inverted Pendulum Using Neural Networks," *IEEE Control Systems Magazine*, vol. 9, no. 3.
- Anderson, M. & Schmidt, D. (1991). "Error Dynamics and Perfect Model Following with Application to Flight Control," *AIAA Journal of Guidance, Control, & Dynamics*, vol. 14, no. 5.
- Åström, K. & Wittenmark, B. (1989). *Adaptive Control*, Addison-Wesley.
- Atkins, S. (1993). *Incremental Synthesis of Optimal Control Laws Using Learning Algorithms*, CSDL Report T-1181, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T.
- Baird, L. & Baker, W. (1990). "A Connectionist Learning System for Nonlinear Control," proceedings, *1990 AIAA Guidance, Navigation, and Control Conference*.
- Baird, L. & Klopff, A. (1992). "Extensions of the Associative Control Process (ACP) Network: Hierarchies and Provable Optimality," proceedings, *2nd International Conference on Simulation of Adaptive Behavior*.
- Baird, L. (1991). *Learning and Adaptive Hybrid Systems for Nonlinear Control*, CSDL Report T-1099, M.S. Thesis, Department of Computer Science, Northeastern University.
- Baker, W. & Farrell, J. (1990). "Connectionist Learning Systems for Control," proceedings, *SPIE OE/Boston '90*.
- Baker, W. & Farrell, J. (1991). "Learning Augmented Flight Control for High Performance Aircraft," proceedings, *1991 AIAA Guidance, Navigation, and Control Conference*.
- Baker, W. & Farrell, J. (1992). "An Introduction to Connectionist Learning Control Systems," in White, D. & Sofge, D., eds., *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold.
- Baker, W. & Millington, P. (1992). "Adaptation & Learning in Control Systems, Application to Flight Control," proceedings, *1992 Government Neural Network Applications Workshop*.
- Baker, W. & Millington, P. (1993). "Design and Evaluation of a Learning Augmented Longitudinal Flight Control System," proceedings, *32nd IEEE Conference on Decision and Control*.

- Barto, A., Sutton, R., & Anderson, C. (1983). "Neuron-Like Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5.
- Battiti, R. (1992). "First and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method," *Neural Computation*, vol. 4, vol. 2, pp. 141-166.
- Berger, T. (1992). *Control of Unmanned Underwater Vehicles Using Spatially Localized Learning Methods*, CSDL-T-1142, S.M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.
- Brumbaugh, R. (1991). "An Aircraft Model for the AIAA Controls Design Challenge," paper no. AIAA 91-2631, presented at the 1991 AIAA Guidance, Navigation, and Control Conference.
- Bryson, A. & Ho, Y. (1975). *Applied Optimal Control*, Hemisphere.
- Bugajski, D., Enns, D., & Elgersma, M. (1990). "A Dynamic Inversion Based Control Law with Application to the High Angle-of-Attack Research Vehicle," proceedings, 1990 AIAA Guidance, Navigation, and Control Conference.
- Calvert, J. (1991). "Development of an Aircraft Simulation Performance Model for Flight Control System Gain Optimization Using Model Following Techniques," draft final report, Naval Air Warfare Center / Aircraft Division / Flight Dynamics & Control Branch, Warminster, PA.
- Cerrato, D. (1993). *Modeling Unknown Dynamical Systems Using Adaptive Structure Networks*, CSDL-T-1194, S.M. Thesis, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology.
- Farrell, J. & Baker, W. (1991). "Learning Augmented Control for Advanced Autonomous Underwater Vehicles," proceedings, 18th Annual AUUVS Technical Symposium and Exhibit.
- Farrell, J. & Baker, W. (1992). "Learning Control Systems," in Antsaklis, P. & Passino, K., eds., *Intelligent and Autonomous Control Systems*, Kluwer Academic.
- Farrell, J. & Baker, W. (forthcoming). "Learning Control Systems: Motivation and Implementation," to appear in *Intelligent Control Systems: Theory and Practice*, Gupta, M. & Sinha, N., eds., IEEE Press.
- Friedland, B. (1986). *Control System Design*, McGraw-Hill.
- Funahashi, K. (1989). "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp. 183-192.
- Gelb, A., ed. (1974). *Applied Optimal Estimation*, MIT Press.
- Gupta, M., ed. (1986). *Adaptive Methods for Control System Design*, IEEE Press.
- Haykin, S. (1991). *Adaptive Filter Theory*, Prentice Hall.
- Hornik, K., Stinchcombe, M., & White, H. (1989). "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks*, vol. 2 pp. 359-366.

- Jacobs, R. (1988). "Increased Rates of Convergence Through Learning Rate Adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295-307.
- Jazwinski, A. (1970). *Stochastic Processes and Filtering Theory*, Academic Press.
- Kailath, T. (1980). *Linear Systems*, Prentice-Hall.
- Kalman, R. (1960). "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME, Journal of Basic Engineering*, March, pp. 35-45.
- Klopf, A. & Morgan, J. (1990). "The Role of Time in Natural Intelligence: Implications of Classical and Instrumental Conditioning for Neuronal and Neural Network Modeling," in Gabriel, M. & Moore, J., eds., *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, MIT Press.
- Klopf, A. (1988). "A Neuronal Model of Classical Conditioning," *Psychobiology*, vol. 16, no. 2, pp. 85-125.
- Klopf, A., Morgan, J., & Weaver, S. (1992). "Modeling Nervous System Function with a Hierarchical Network of Control Systems that Learn," proceedings, 2nd *International Conference on Simulation of Adaptive Behavior*.
- Kreisselmeier, G. (1980). "Perspectives on the Application of Adaptive Control to Aircraft Systems," in Narendra, K. & Monopoli, R., eds., *Applications of Adaptive Control*, Academic Press.
- Livstone, M., Farrell, J., & Baker, W. (1992). "A Computationally Efficient Algorithm for Training Recurrent Connectionist Networks," proceedings, 1992 *American Control Conference*.
- Maciejowski, J. (1989). *Multivariable Feedback Design*, Addison-Wesley.
- Maybeck, P. (1979). *Stochastic Models, Estimation, and Control: Volume I*, Academic Press.
- McRuer, D., Ashkenas, I., & Graham, D. (1973). *Aircraft Dynamics and Automatic Control*, Princeton University Press.
- Michie, D. & Chambers, R. (1968). "BOXES: An Experiment in Adaptive Control," in Dale, E. & Michie, D., eds., *Machine Intelligence 2*, Oliver & Boyd.
- MIL-F-8785C (1980). *Military Specification, Flying Qualities of Piloted Airplanes*.
- MIL-STD-1797A (1990). *Military Standard, Flying Qualities of Piloted Airplanes*.
- Millington, P. & Baker, W. (1992). *Learning Augmented Flight Control for High Performance Aircraft*, CSDL-R-2508, Final Report, USN Contract No. N62269-91-C-0033, Draper Laboratory, Cambridge, MA.
- Millington, P. (1991). *Associative Reinforcement Learning for Optimal Control*, CSDL Report T-1070, S.M. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.
- Millington, P., Baker, W., & Koenig, M. (1993). "Control Augmentation System (CAS) Synthesis via Adaptation & Learning," proceedings, 1993 *AIAA Guidance, Navigation, and Control Conference*.

- Moody, J. & Darken, K. (1989). "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, vol. 1, no. 2, pp. 281-294.
- Morgan, J., Patterson, E., & Klopff, A. (1990). "Drive-Reinforcement Learning: A Self-Supervised Model for Adaptive Control," *Network: Computation in Neural Systems*, vol. 1, pp. 439-448.
- Narendra, K. & Annaswamy, A. (1989). *Stable Adaptive Systems*, Prentice-Hall.
- Narendra, K. & Monopoli, R., eds., (1980). *Applications of Adaptive Control*, Academic Press.
- Narendra, K., Ortega, R., & Dorato, P., eds. (1991). *Advances in Adaptive Control*, IEEE Press.
- Nistler, N. (1992). *A Learning Enhanced Flight Control System for High Performance Aircraft*, CSDL Report T-1127, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T.
- Parkinson, G. & Smith, J. (1964). "The Square Prism as an Aeroelastic Non-linear Oscillator," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 17, pp. 225-239.
- Poggio, T. & Girosi, F. (1990). "Networks for Approximation and Learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481-1497.
- Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (1988). *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press.
- Ramesh, A., Senol, U., & Garba, J. (1989). "Computational Complexities and Storage Requirements of Some Riccati Equation Solvers," *AIAA Journal of Guidance, Control, & Dynamics*, vol. 12, no. 4.
- Rumelhart, D., Hinton, G., & Williams, R. (1986). "Learning Internal Representations by Error Propagation," in Rumelhart, D. & McClelland, J., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press / Bradford.
- Samuel, A. (1959). "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210-229; reprinted in Feigenbaum, A. & Feldman, J., eds., *Computers and Thought*, McGraw-Hill.
- Slotine, J. & Li, W. (1991). *Applied Nonlinear Control*, Prentice-Hall.
- Sorenson, H., ed. (1985). *Kalman Filtering: Theory and Application*, IEEE Press.
- Stein, G. (1980). "Adaptive Flight Control: A Pragmatic View," in Narendra, K. & Monopoli, R., eds., *Applications of Adaptive Control*, Academic Press.
- Steinberg, M. (1992). "Potential Role of Neural Networks and Fuzzy Logic in Flight Control Design and Development," AIAA paper no. 92-0999, 1992 AIAA Aerospace Design Conference.
- Stevens, B. & Lewis, F. (1992). *Aircraft Control and Simulation*, John Wiley & Sons.

- Sutton, R. (1984). *Temporal Credit Assignment in Reinforcement Learning*," Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA.
- Sutton, R. (1988). "Learning to Predict by the Methods of Temporal Differences," *Machine Learning 3*, Kluwer Academic.
- Tsytkin, Y. (1973). *Foundations of the Theory of Learning Systems*, Academic Press.
- Vidyasagar, M. (1985). *Control System Synthesis: A Factorization Approach*, MIT Press.
- Vos, D., Baker, W., & Millington, P. (1991). "Learning Augmented Gain Scheduling Control," proceedings, 1991 AIAA Conference on Guidance, Navigation, and Control.
- Watkins, C. (1989). *Learning from Delayed Rewards*, Ph.D. Thesis, Cambridge University, Cambridge, England.
- Widrow, B. & Hoff, M. (1960). "Adaptive Switching Circuits," 1960 WESCON Convention Record, Part IV, pp. 96-104.
- Widrow, B., McCool, J., Larimore, M., & Johnson, C. (1976). "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter," *Proceedings of the IEEE*, vol. 64, no. 8.
- Youcef-Toumi, K. & Ito, O. (1990). "A Time Delay Controller for Systems with Unknown Dynamics," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 112.

ATTACHMENT 1

Reprint of:

Baird, L. (1991). *Learning and Adaptive Hybrid Systems for Nonlinear Control*, CSDL Report T-1099, M.S. Thesis, Department of Computer Science, Northeastern University.

**LEARNING AND ADAPTIVE HYBRID SYSTEMS
FOR NONLINEAR CONTROL**

Leemon C. Baird III

Submitted to the Department of Computer Science
May, 1991 in partial fulfillment of the requirements for the
Degree of Master of Science in Computer Science

ABSTRACT

Connectionist learning systems may be considered to be automatic function approximation systems which learn from examples, and have received an increase in interest in recent years. They have been found useful for a number of tasks, including control of multi-dimensional, nonlinear, or poorly modeled systems. A number of approaches have been applied to control problems, such as modeling inverse dynamics, backpropagating error through time, reinforcement learning, and dynamic programming based algorithms. The question of integrating partial *a priori* knowledge into these systems has often been a peripheral issue.

Control systems for nonlinear plants have been explored extensively, especially approaches based on gain scheduling or adaptive control. Gain scheduling is the most commonly used in practice, but often requires extensive modeling or manual tuning, and is susceptible to modeling uncertainty and time-varying dynamics. Adaptive control addresses these problems, but usually cannot react to known spatial dependencies (nonlinearities) quickly enough to compete with a well-designed gain scheduled system.

This thesis explores a hybrid control approach that uses a connectionist learning system to store spatial dependencies discovered by an indirect adaptive controller. The connectionist system learns to anticipate the parameters estimated by the indirect adaptive controller, effectively becoming a gain scheduled controller. The combined system is then able to exhibit some of the advantages of gain scheduled and adaptive control, without the extensive manual tuning required by traditional methods. Subsequently, a technique is presented for making use of input/output partial derivative information from the network. Finally, the applicability of second-order learning methods to control is considered, and areas of future research are suggested.

| | |
|--------------------|-------------------------------|
| Thesis Supervisor: | Dr. Ronald J. Williams |
| Title: | Professor of Computer Science |
| Thesis Supervisor: | Mr. Walter L. Baker |
| Title: | Technical Staff, CSDL |

ATTACHMENT 1

ACKNOWLEDGMENT

This report was prepared at The Charles Stark Draper Laboratory, Inc. with support provided by the U.S. Air Force under Contract F33615-88-C-1740. Publication of this report does not constitute approval by the sponsoring agency of the findings or conclusions contained herein. It is published solely for the exchange and stimulation of ideas.

ATTACHMENT 1

TABLE OF CONTENTS

| | | |
|-------|--|-----|
| 1 | INTRODUCTION | 143 |
| 1.1 | Motivation | 143 |
| 1.2 | Problem Description | 144 |
| 1.3 | Thesis Objectives and Overview | 146 |
| 2 | BACKGROUND | 147 |
| 2.1 | Connectionist Learning Systems | 147 |
| 2.1.1 | Single-Layer Networks | 149 |
| 2.1.2 | Multilayer Networks | 154 |
| 2.2 | Traditional Control | 158 |
| 2.2.1 | Bang-Bang Control | 160 |
| 2.2.2 | Proportional Control | 160 |
| 2.2.3 | PID Control | 161 |
| 2.2.4 | Adaptive Control | 162 |
| 2.2.5 | Gain Scheduled Control | 163 |
| 2.3 | Connectionist Learning Control Approaches | 164 |
| 2.3.1 | Producing Specified Control Signals | 165 |
| 2.3.2 | Following Specified Trajectories | 166 |
| 2.3.3 | Optimizing Specified Reinforcement Signals | 172 |
| 3 | HYBRID CONTROL ARCHITECTURE | 179 |
| 3.1 | The Learning Component | 182 |
| 3.2 | The Adaptive Component | 183 |
| 3.3 | The Hybrid System | 184 |
| 3.4 | Derivation of the Hybrid With Known Control Effect | 185 |
| 3.5 | Derivation of the Hybrid With Unknown Control Effect | 188 |

ATTACHMENT 1

| | | |
|-------|--|-----|
| 4 | LEARNING SYSTEMS USED | 192 |
| 4.1 | Backpropagation Networks | 192 |
| 4.2 | Delta-Bar-Delta | 200 |
| 5 | EXPERIMENTS | 203 |
| 5.1 | The Cart-Pole System | 205 |
| 5.2 | Organization of the Experiments | 212 |
| 5.3 | Mid-Trajectory Spatial Nonlinearities | 213 |
| 5.4 | Trajectory-End Spatial Nonlinearities | 218 |
| 5.5 | Trajectory-Start and Trajectory-End Nonlinearities | 222 |
| 5.6 | Noise and Nonlinear Functions of Control | 230 |
| 5.7 | Comparison of Connectionist Networks Used | 235 |
| 5.7.1 | Sigmoid | 235 |
| 5.7.2 | Sigmoid With a Second-Order Method (Delta-Bar-Delta) | 238 |
| 6 | CONCLUSIONS AND RECOMMENDATIONS | 241 |
| 6.1 | Summary and Conclusions | 241 |
| 6.2 | Recommendations for Future Work | 242 |
| | BIBLIOGRAPHY | 243 |

1 INTRODUCTION

1.1 MOTIVATION

The design of effective automatic control systems for nonlinear plants presents a difficult problem. Because direct analytic solutions to such problems are generally unobtainable, various approximate solution methods must be used (e.g., gain scheduling). The design problem is further complicated by modeling errors. If there are significant plant dynamics that are not included in the design model, or if the plant dynamics change unpredictably in time, then the closed-loop system can perform worse than expected and may even be unstable. Furthermore, if the sensors are noisy, then filters will be required, which tend to make the control system slow to recognize changes in the plant (from either unmodeled or time-varying dynamics).

Traditional gain scheduled controllers often require extensive manual tuning to design and develop, and do not deal well with unmodeled spatial dependencies, disturbances, or time-varying plants. Adaptive controllers can handle these difficulties in principle, but in practice may adapt to spatial dependencies so slowly that the controller is not as good as a gain scheduled controller would be.

In contrast, an "intelligent" controller operating in a complex environment should be able to accommodate a certain degree of uncertainty (e.g., from time-varying dynamics, noise, and disturbances). More importantly, it should be able to learn from experience to anticipate previously unknown, yet *predictable*, effects (e.g., quasi-static nonlinearities). A possible solution to this problem might be a hybrid adaptive / learning control system which could both adapt to disturbances and learn to anticipate spatial nonlinearities.

1.2 PROBLEM DESCRIPTION

Traditional adaptive control tends to be inefficient and perform poorly with respect to significant, unmodeled spatial dependencies, while traditional gain scheduled control has difficulty with poorly modeled dynamics. The problem is to find a system that can control a plant in the presence of both simultaneously, while incorporating incomplete and possibly erroneous (but not debilitating) *a priori* knowledge of the system.

Sometimes a controller is required which can force a plant to follow some desired reference trajectory. This *model reference* control problem is approached here using both traditional control techniques and learning systems. The approaches explored do not require that the reference trajectory satisfy any special constraints, such as being generated by a linear system. The only requirement is a well-defined method for calculating at each point in time the desired rate of change of the plant state.

Few assumptions are made about the plant itself; it can be nonlinear, poorly modeled, and subject to unpredictable disturbances. The sensor readings from the plant must contain sufficient information to observe its state and control it, but may be noisy and otherwise incomplete. For example the plant may have actuator dynamics involving internal state within the actuators that is not measured by any sensor. Specifically, it can have unknown dynamics that are functions of both state and time. The plant can have *spatial dependencies*, that is nonlinearities that are functions of state and are either static or quasi-static in time. In addition, it can also have *temporal dependencies* which are functions of time, caused by disturbances and other short-term, unpredictable events.

Another important property of the control system is that it be possible to incorporate any *a priori* knowledge. This should include knowledge about both the behavior of the plant in the absence of any control signals, and the effect of the control signals on the plant. Moreover, it is especially important that errors in the *a priori* information not cripple the

ATTACHMENT 1

controller in the long run. The controller should be able to eventually learn these errors and compensate for them.

Various algorithms for connectionist learning systems are often proposed and compared on very small "toy" problems. The error to be minimized is usually defined as the total squared output error, summed over the output for each training example. The problems arising in learning control often do not resemble these test problems, and so it is difficult to predict how various proposed modifications will affect learning controllers. The problems in control typically involve learning functions that map continuous inputs to continuous outputs, and these functions are generally smooth with possibly a few discontinuities. For a control problem, the error is defined as the total squared error, integrated over the entire domain. Learning systems that can quickly learn to fit a function to a small number of points may not be able to quickly learn the continuous functions arising in typical control problems.

Another important aspect of learning control is the order in which training examples become available. Most proposed learning systems are tested on learning problems involving a fixed set of training examples, which are all available at the same time, and which can be accessed in any order. In control problems, the plant being controlled may change its state slowly, or tend to spend large amounts of time in a small regions of the state-space (e.g., near an operating point). This may cause the learning system to receive a large number of similar training examples before seeing different training examples. For some learning systems this uneven ordering of training data may not matter. For others, it may cause the system to learn more slowly or to forget important information. In any case, this is an aspect of learning control that must be taken into account when comparing various learning systems for use in a controller.

1.3 THESIS OBJECTIVES AND OVERVIEW

The object of this thesis is to find methods for combining learning systems with adaptive systems in order to achieve good control in the presence of both spatial and temporal functional dependencies. Several methods are developed for augmenting the estimation carried out by an indirect adaptive system with the additional information available from a learning system. In addition to developing this *learning augmented estimation*, various issues in the construction and use of connectionist learning systems are explored in this context.

Chapter 2, *Background*, gives some of the important concepts and historical development of connectionist systems, control systems, and approaches to using connectionist systems for control.

Chapter 3, *Hybrid Control Architecture*, covers the adaptive controller and connectionist networks that are integrated into a single hybrid controller. Both the individual components and the final, integrated system, are motivated from current problems, and are described in detail.

Chapter 4, *Connectionist Learning for Control*, covers some of the difficulties associated with learning systems for control, and describes the methods used here to deal with those difficulties.

Chapter 5, *Experiments*, describes the various simulations performed and their results. These results are presented graphically and are interpreted in relation to the original goals.

Chapter 6, *Conclusions and Recommendations*, summarizes what has been accomplished, draws conclusions, and points out areas in which future research should be focused.

The bibliography lists those works which were used in the preparation of this thesis, together with other, related works.

2 BACKGROUND

The hybrid learning / adaptive controller combines connectionist learning systems with traditional control systems, and modifies each of these components to improve the ability of the hybrid to combine the strengths of each. Before describing the hybrid system itself, it is first necessary to cover some of the important developments and concepts relating to these components. Section 2.1 covers the development of some of the important ideas in connectionist learning systems, and Section 2.2 deals with some of the common approaches in traditional control theory. Finally, Section 2.3 describes some of the approaches that have been taken in building learning controllers or incorporating connectionist learning systems into control systems.

2.1 CONNECTIONIST LEARNING SYSTEMS

The application of connectionist learning systems to problems in control has received considerable attention recently. Such systems, usually in the form of feedforward multilayer networks, are appealing because they are relatively simple in form, can be used to realize general nonlinear mappings, and can be implemented in parallel computational hardware. An example of a simple network is shown in figure 2.1. The network consists of nodes and connections between nodes. A node may have several real-valued inputs, each of which has an associated connection weight (also real-valued). Each node computes a nonlinear function of the weighted sum of its inputs, and then sends the result out along all the connections leaving the node. Nodes are arranged in layers, with nodes in each layer sending outputs only to nodes in subsequent layers. In such feedforward networks, it is easy to calculate network outputs, given a set of inputs.

ATTACHMENT 1

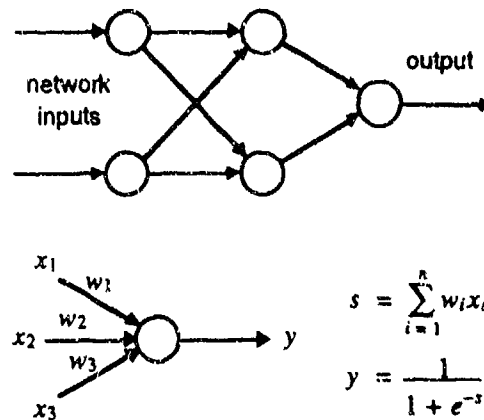


Figure 2.1 A connectionist network

A key feature of feedforward multilayer networks is that any piecewise smooth function can be approximated to any desired accuracy by some arbitrarily large network having the appropriate weights [HW89]. Given the correct weights, a network can be used to implement a nonlinear function that is useful for a control application. The difficulty is in finding the appropriate weights. No known algorithm guarantees finding satisfactory weights for all layers of a multilayer network, and Minsky and Papert pointed out in 1969 that the small networks networks that are guaranteed to converge do not scale well for some large problems [MP69]. Many saw this as an indication that connectionist approaches were not useful in general.

One event that helped change this perception was the development of the error Backpropagation algorithm, independently developed by Werbos [Wer74], Parker [Par82], LeCun [LeC87], and Rumelhart, Hinton, and Williams [RHW86]. Error back-propagation is a gradient descent algorithm that modifies network weights incrementally to minimize a particular measure of error. The error is usually defined as the sum of the squared error in the output over the set of inputs. The network functions are continuously differentiable, so it is possible to calculate the gradient of the total error with respect to the weights, and to adjust the weights in the direction of the negative gradient. As with all gradient descent

optimization techniques, there exists a possibility of converging to a non-optimal local minimum. Despite this, learning systems using back-propagation have been shown to find good solutions to various real world problems including difficult, highly nonlinear control problems. No difficulties due to the presence of local minima were observed in any of the experiments that are described in this thesis.

Backpropagation and many other connectionist learning algorithms tend to converge slowly, and so are more useful for learning quasi-static nonlinear functions than for adapting to rapidly changing functions.

2.1.1 Single-Layer Networks

The earliest connectionist systems were *single-layer* networks. Single-layer networks are networks that implement functions with the property that the function is a linear combination of other functions, and only the weighting factors in that linear combination change during learning. These networks tend to be less powerful, but the learning rules are simpler, and so these architectures received the earliest attention.

Perceptrons

One of the early connectionist network models was the simple perceptron, developed by Rosenblatt [Ros62] in the late 50's (as discussed in [RZ86][Sim87]). Rosenblatt coined the term *perceptron* to refer to connectionist systems in general, including those with multiple layers and feedback. He is most widely known for the development of the *simple perceptron*. A simple perceptron is a device which takes several inputs, multiplies each one by an associated integer called its weight, and finds the sum of these products. The simple perceptron has a single output, and the inputs and output are each 1 or -1. The output is -1 if the weighted sum of the inputs is negative, and 1 if the sum is nonnegative.

If the input is thought of as a pattern and the output as a truth value, then the simple

ATTACHMENT 1

perceptron can be thought of as a classifier which determines whether or not inputs belong to a given class. Given a set of input patterns along with their correct classification, it is sometimes possible to find weights that will cause a simple perceptron to classify those patterns correctly. Specifically, if such a set of weights exists, then Rosenblatt proved that a very simple algorithm will always succeed in finding those weights, learning only from presentations of inputs and their correct classifications. The algorithm simply started with arbitrary weights, and repeatedly classified training examples. Whenever it got a classification wrong, each weight that had an effect on the result was incremented or decremented by one, so as to make the resulting sum closer to the correct answer. Rosenblatt's "perceptron learning theorem" proving the validity of this algorithm is one of the more influential results of his research.

It is helpful to think of the inputs to the network as a vector representing a point in some high-dimensional space. The weighted sum of the inputs is a hyperplane in that space, and the output from the simple perceptron will classify inputs based on which side of the hyperplane they lie on. This means that a single simple perceptron is only capable of classifying inputs into one of two linearly separable sets, sets which can be separated by a hyperplane. Although this limits the power of a single simple perceptron, it is still useful to know that any such classification can be learned simply by training the simple perceptron with examples of correct classifications.

This limitation on the power of perceptrons can be overcome if the outputs of several simple perceptrons feed in to another simple perceptron, thus forming a multilayer perceptron. Rosenblatt was able to show that for any arbitrary desired classification of the input patterns, there exists a two-layer perceptron which can act as a perfect classifier for that mapping. Unfortunately, there is no known learning algorithm that is guaranteed to find the correct weights for a multilayer perceptron as there was in the case of the single-layer perceptron. Minsky and Papert, in their 1969 book *Perceptrons* [MP69], analyzed single-layer perceptrons and pointed out a number of difficulties with them. Simple

ATTACHMENT 1

perceptrons are only able to recognize linearly separable classes, and so cannot calculate an exclusive OR, or recognize whether the set of black bits in a picture is connected or not. The problem remains even if the inputs to the perceptron are arbitrary functions of proper subsets of the input pattern. Despite the interesting features of single-layer perceptrons, their conclusion was that "there is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile" [MP69]. Minsky later considered *Perceptrons* to be overkill, an understandable reaction to excess hyperbole which was diverting researchers into a false path [RZ86]. However at the time, the book was one of the factors contributing to a decrease in interest in connectionist models in general.

Samuel's Checker Player

Another early system was Samuel's checker playing program [Sam59][Sam67]. This was the first program capable of playing a nontrivial game well enough to compete well with humans, and it was an important system because it introduced a number of new ideas. It used both book (table) lookup and game-tree searches, and was the first program in which the now common procedure of alpha-beta pruning was used. It also had a learning component which was not referred to as a neural network or connectionist system at the time, but which strongly resembles many such systems.

The program chose its move in checkers by searching a game-tree to some depth and picking the best move. Alpha-beta pruning and other subtleties were used to make the search more efficient, but the basic component needed to make it work was a function that could compare the desirability of reaching each of several possible board positions. Given an exhaustive search, this scoring function could be as simple as "choose a move that ensures a win if possible; otherwise avoid a loss." Since Samuel could only search a small number of moves, the scoring function was very important, and so he built it to combine

ATTACHMENT 1

the best *a priori* knowledge he could find with additional knowledge found by the program through learning.

The *a priori* knowledge which Samuel started with was a set of heuristic functions derived from a knowledge of what good human players consider important. For example, one such function was the number of pieces each player had on the board; another was how many possible moves the computer had available to choose between. Each of these functions were hand built to have a good chance of being significant, to be quick and easy to calculate, and to return a single number instead of a vector or a symbol. The scoring function was simply a linear combination of each of the outputs of these functions. Samuel referred to this linear function as a polynomial. The learning system was designed to pick the functions that would be included in the linear combination, and to pick weights for these functions.

All of the weights were initially set to arbitrary values. The program could then play games against a copy of itself, where only one of the two copies would learn during a given game. The score for a board position represented the expected outcome of the game. If the score on the next turn was different, then the later score can be assumed to be more accurate than the earlier score, since it is based on looking farther ahead in the game. Therefore the weights would all be modified slightly so that the earlier score would more nearly match the later score. The polynomial had some fixed terms that were never changed by learning, which ensured that the score of a board at the end of the game would always be accurate, preferring wins to losses. The process described here is very similar to how the perceptron learned, changing weights slightly on each time step so as to decrease error. There were other important aspects of Samuel's algorithm beyond this, such as occasionally randomly changing the function to escape local minima, but the core of the learning process was this simple hill climbing algorithm.

Although Samuel said he was avoiding the "Neural-Net Approach" in his program by including *a priori* information and learning rules specific to games, the ideas which he

ATTACHMENT 1

developed are similar in many ways to much later systems for multilayer networks, optimal control, and reinforcement learning described below. His ideas influenced the work of Michie and Chambers' Boxes [MC68] and Sutton's Temporal Difference (TD) and Dyna learning [Sut88][Sut90][BSW89][BS90]. Samuel's algorithm can actually be seen as a type of incremental dynamic programming [WB90].

ADALINE and MADALINE

A third system which was developed in the late 1950's was Widrow's ADALINE and MADALINE [Wid89]. He developed a type of adaptive filter which is still in widespread use today in such items as high-speed modems. It worked by multiplying several signals by weights, summing them, looking at the output, and then adjusting the weights according to the errors in the output. His training data was analog and noisy and came from changing signals, but for the most part his filters were similar to the perceptrons or polynomial scoring functions described above. When weights were changed in proportion to their effect on the error, and when the changes became smaller over time, Widrow proved that the weights were guaranteed to converge. He then went on to add a squashing function to the output of one of his filters, forcing the output to +1 or -1 on each time step, and used it for pattern recognition. This "Adaptive Linear Neuron" (ADALINE) [Wid89] was then built in actual hardware, where weights were represented by the electrical resistance of copper coated graphite rods, and learning was accomplished by causing more copper to come out of solution and plate the rods. When the the outputs of multiple ADALINE's were fed into another ADALINE, this formed what Widrow called a MADALINE (for multiple ADALINES). By doing this, he was able to get around the problem of only learning linearly separable functions. However, he did not have a method for training the weights that connected the first set of ADALINE's to the last one, so he simply fixed all the weights at a value of one.

2.1.2 Multilayer Networks

As can be seen in the above descriptions, a number of researchers were developing very similar systems in the late 50's and early 60's, some of which generated a great deal of excitement. The particular difficulties pointed out in *Perceptrons* could not be overcome as long as the output of the device was simply a function of a linear combination of the inputs. A second layer needed to be added that would take its inputs from the outputs of the first layer. Widrow added a second layer in the MADALINE, but was unable to train all of the weights. The problem of multilayer learning was one of the reasons that interest in connectionism tended to wane until its resurgence in the late 80's.

Hebbian Learning

In 1949, Hebb proposed a simple model of learning based on his studies of biological neurons. A neuron in this model would generate an output that was some function of the weighted sum of its inputs. Unlike the models described above, these weights would learn without any external training signal at all. The learning occurred according to the Hebbian Learning Rule, which stated that the efficacy of a plastic synapse increased whenever the synapse was active in conjunction with activity of the postsynaptic neuron. This meant that the weight of a connection increased whenever both connected neurons had high outputs at approximately the same time, and decreased when only one of them did.

Drive-Reinforcement

The basic Hebbian model has been refined in various ways over the years to improve both its ability to model animal behavior, and its ability to perform useful functions in systems such as controllers. One important development in this line of research is Klopff's drive-reinforcement model [Klo88]. In this model, three major modifications are made to the basic Hebbian model.

ATTACHMENT 1

First, instead of correlating the output of one neuron with the output of another, the correlation is made between *changes* in outputs. If signal levels are thought of as drives, such as hunger, then it does not make sense for the network to change weights merely on the basis of the existence of these drives. However, when a signal level changes, such as would happen when hunger is relieved by eating, or pain is increased due to damage being done to an animal, then the network should change. The second modification is to correlate past inputs (or changes in inputs) with current outputs (or changes in outputs). This generally allows the network to learn to predict, which a purely Hebbian network is unable to do. The third modification is to always modify weights in proportion to the current weight value. This causes learning to follow an "S" shaped curve. At first, a given weight increases slowly, then grows more rapidly, and finally slows down again and approaches an asymptotic value. This result is more consistent with the result of experiments with learning in animals.

This model has proven accurate in reproducing a wide range of actual animal learning experiments. For example, it is possible to simulate Pavlov's results in classical conditioning. A single neuron can be given one input representing the ringing of a bell, and another input representing the taste of meat juice. If the output of the neuron is interpreted as the salivation response of Pavlov's dogs, then the system can be seen to slowly become classically conditioned, learning to salivate in response to the bell with an "S" shaped curve. When the meat juice stimulus is removed, it demonstrates extinction of the response in a manner which is also realistic.

Drive-reinforcement learning has also been applied to control. Multiple drive-reinforcement neurons have been connected with other components to form controllers for traditional control problems, as well as for the problem of traversing a maze to the reward at the end. This is especially interesting in light of the fact that each individual neuron is not trying to explicitly minimize an error, as in the other controllers discussed here.

Backpropagation

One of the major contributing factors to the return of widespread interest in connectionist systems is the development of the Error Backpropagation algorithm. The basic idea is simple. A network consists of a set of inputs, a set of outputs, and a set of nodes which calculate an output as a function of inputs. The nodes are arranged in layers, with the inputs connecting to the first layer and the last layer connecting to the outputs. The network is *feedforward*, i.e., the complete directed graph of nodes and connections is acyclic.

Each node functions by taking each of its inputs, multiplying it by an associated weight, taking a smooth, monotonic function of the sum (such as the hyperbolic tangent), and then sending the result to all of its outputs. If the network is presented with a set of different inputs, it will generate an output for each one. The total squared error in the outputs J can then be calculated, and the weights w changed according to:

$$J = \sum_{i=1}^n (f(x_i, w) - d_i)^2$$

$$\Delta w_i = -\alpha \frac{\partial J}{\partial w_i}$$

where:

J = total error for network with weights w

n = number of training examples

α = learning rate (controlling step size)

x_i = input to network for i th training example

d_i = desired output from network for i th training example

$f(x_i, w)$ = actual output from network for i th training example

The change in each weight is proportional to the associated partial derivative. In a multilayer network, the output of each layer is a simple function of the output of the layer before it. This allows all of the partial derivatives to be calculated quickly by starting at the output of the network and working backward according to the chain rule. Propagating

errors backward requires as little computation as propagating the original signals forward. Furthermore, the error calculations can all be done locally, in the sense that information need only flow back through the network along the connections that already exist between nodes. These properties combine to make Backpropagation powerful yet low in both computation time and hardware required.

This gradient descent process is simple and works well for multilayer networks, but is not guaranteed to find the best weights possible. As with all hill-climbing methods, it can get stuck in a local minimum. Although this method cannot be guaranteed to find the correct answer (as simple perceptrons were), it is still a useful method which has been shown to work well on a variety of problems. Unfortunately, pure gradient descent methods often converge slowly in the presence of "troughs" in the error surface. If the error as a function of network weights is thought of as a high-dimensional surface, then a long, thin trough in this surface slows convergence. If the current set of weights is a point on the side of a trough, then the gradient will point mainly down the side of the trough, and only slightly in the direction along the trough toward the local minimum. If the weight changes in large steps, it will oscillate across the trough. If it changes in small steps, then it converges to the local minimum very slowly.

There are a number of approaches to speeding up convergence in this case. One is to look at the second derivative in addition to the gradient at each point. If a network has one output and multiple weights, then the second derivative is a matrix giving the second partial derivative of the output with respect to each possible pair of weights. This matrix, called the Hessian, has a useful geometric interpretation. Multiplying a vector by this matrix stretches the vector in some directions and compresses it in others. For the direction in which the error surface has least curvature, the Hessian will compress vectors. For the direction in which the error surface has greatest curvature, the Hessian will stretch vectors. Multiplying a vector by the inverse of the Hessian has the opposite effect. Multiplying the gradient by the inverse of the Hessian will cause the weights to change more in the

direction along a trough (where the curvature is small), and less across the trough (where the curvature is large). If the network has N weights, this requires inverting an N by N matrix on every iteration during training. This overwhelming flood of calculations may defeat the purpose by requiring more computation than is saved by the shorter path to convergence. This is why a number of approximate approaches have been proposed for solving this problem, such as using only the diagonal of this matrix, or using heuristics that approximate the effect of the inverse Hessian.

2.2 TRADITIONAL CONTROL

Control theory deals with the problem of forcing some system, called the *plant*, to behave in desired manner. The set of relevant properties of the plant which change through time is called the *state*, and is represented by the real vector \mathbf{x} . For example, in the cruise control for a car, the state might include the current speed and slope of the ground. If the state cannot be measured directly, then the sensor readings are represented by another real vector \mathbf{y} . The *control action* is the set of signals applied to the plant by the controller, and is represented by the real vector \mathbf{u} . The plant state then is assumed to evolve in time according to:

$$\begin{aligned}\dot{\mathbf{x}}_t &= \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \\ \mathbf{y}_t &= \mathbf{g}(\mathbf{x}_t)\end{aligned}$$

The majority of control theory is devoted to the special case where the plant is *linear*, in which case the state evolves according to

$$\begin{aligned}\dot{\mathbf{x}}_t &= \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t \\ \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t\end{aligned}$$

where \mathbf{A} , \mathbf{B} , and \mathbf{C} are constant matrices. Even if a plant is not truly linear, it is often close enough to linear within certain regions of the state-space that a controller can be designed for that region based on a linear approximation of the plant. This is useful since the theory for linear plants is better developed than for nonlinear plants [D'A88].

Once the plant has been modeled, the controller must be designed to accomplish some purpose. If the goal is to keep the state at a certain value, then the controller is called a *regulator*. If the goal is to force the plant to follow a given trajectory with a specified transient response, then the controller is a *model reference tracking* controller. If the goal is to minimize some cost function of the whole trajectory, then it is an *optimal control* problem.

Traditional control techniques are based on approaches such as bang-bang, proportional, proportional-integral-derivative (PID), gain scheduling, and adaptive control, each described in a section below. These are important control approaches with which connectionist control techniques should be compared. In addition to this, most of them can be included, directly or indirectly, in the hybrid system developed in this thesis.

Several of the systems described here were first demonstrated on a standard *cart-pole* system problem. This plant is illustrated in figure 2.2.

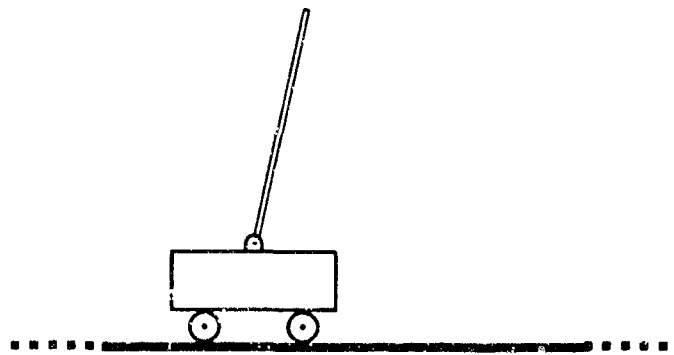


Figure 2.2 The cart-pole plant

In this problem, the cart is confined to a one dimensional track, and force can be applied to it in either direction to cause it to move left or right. On top of the cart is a pole, which is hinged at the bottom and can swing freely. No forces are applied to the pole directly, so it is only influenced indirectly through forces applied to the cart. The problem of balancing the pole is similar to the problem of balancing a broomstick on a person's hand. This is a standard control problem and is useful for demonstrating new control

methods. A version of this problem is used here to test the new hybrid control systems developed in this thesis.

2.2.1 Bang-Bang Control

The simplest form of control is a controller that only has two possible outputs. This "bang-bang" control is commonly used in thermostats which alternate between running the heating system full on and turning it off completely. This type of control has also been used in a learning system to balance a pole on a cart while keeping the cart within a certain region [BSA83]. Unfortunately, bang-bang control systems are generally incapable of exercising very fine control, and so usually lead to *limit cycles* in the plant being controlled, i.e. the state repeatedly follows a certain periodic path instead of settling down to a single state. A pole can actually be balanced on a cart by always applying a certain force in the same direction the pole is leaning. Naturally, this leads to a limit cycle with the pole swinging back and forth between two extremes. For finer control, a more general controller is required, such as a proportional controller.

2.2.2 Proportional Control

A proportional controller is perhaps the simplest controller imaginable that still has continuously varying control actions. Each input to the controller is a real value, representing one element of the output of the plant being controlled. In a regulator, that is the only input, and the controller tries to control the plant so that all of the elements of the state vector are nominally zero. In a general controller, each element of the desired state vector is also an input. The controller then multiplies each input by a constant gain, possibly adds a constant, and uses the result as the control signal. If the control action is a vector involving several signals, then the same process is followed for each of them, using a different set of gains each time.

To design a satisfactory proportional controller, it is first necessary to have a good

model of the system being controlled. If the plant is linear and perfectly modeled, or even if the plant is only close to linear, then it is often possible for a proportional controller to do an acceptable job of controlling it.

2.2.3 PID Control

If the control signal to a plant is simply proportional to the error in its state, then as the state approaches the desired state, the correcting force will decrease proportionally. Often, there will be some point near the desired point at which the small correcting force is balanced by other forces, and the plant will settle into a steady state which has a slight error. In order to overcome this steady state error, the controller might integrate the error over a long period of time, and add a component to the control signal proportional to this integral. It may also be possible to improve the control signal by taking into account not only the output error, but also how the output error is changing in time. For this reason it may be useful to add a term to the control signal proportional to the derivative of the output error.

If both of these modifications are made to a proportional controller, it is then called a proportional plus integral plus derivative (PID) controller. If the input to this controller and the output from it are considered as functions of time and the Laplace transform of them is taken, then the relationship between input and output is simple. It is some quadratic function of s divided by s . In discrete-time control, this means that the output of the controller is a linear combination of four things: the control actions on the previous time step, the current output error, the output error on the previous time step, and the output error on the time step before last. Since the control output is at least partially proportional to the control applied on the previous time step, a small error in plant output causes the controller output to keep increasing until the error is gone. This is the integral portion of the controller. Since plant output errors from three different time-steps are used, it is possible to subtract them and estimate how fast the output error is changing. This is the

derivative aspect of the controller. Also, since the current output error affects the controller output directly, it has a proportional control component. Therefore all three types of control are present, and the controller is referred to as a PID controller.

PID control is widely used; in fact perhaps 90% of all of the controllers in existence are PID controllers (or PI or P, which are just PID with some gains set to zero) [Pal83]. If a plant is linear, it is often possible to design PID controllers that give the desired performance. If a plant is nonlinear, but will usually stay in some small region of its state-space, then it is often practical to approximate the plant with a linear model in that region and design a PID controller for that model. This model can be derived from the full, nonlinear equations describing the plant, by taking the derivative of those equations, and evaluating it at a given point in the middle of the region of interest.

2.2.4 Adaptive Control

Instead of creating a fixed controller based on *a priori* knowledge of a plant, it is sometimes beneficial to build a controller that can change if the plant is different than the model, or if the plant changes or experiences disturbances. Starting in the early 1950's, researchers enthusiastically pursued adaptive control, especially for aircraft, but without much underlying theory. Interest then diminished in the early 1960's due to a lack of theory and a disaster during a flight test [Åst83]. More recently, adaptive control is finally beginning to reemerge as a more widely used approach.

Adaptive control techniques can be categorized as either indirect adaptive control or direct adaptive control. Indirect adaptive control utilizes an explicit model of the plant, which is updated periodically, to synthesize new control laws. This approach has the important advantage that powerful design methods (including optimal control techniques) may be used on-line; however, it has the key disadvantage that on-line model identification is required. Alternatively, direct adaptive control does not rely upon an explicit plant model, and thus avoids the need to perform model identification. Instead, the control law

is adjusted directly, based on the observed behavior of the plant. In either case, the controller will adapt if the plant dynamics change by a significant degree.

Adaptive controllers are usually designed with the assumption that there is some modeling error, but that the plant behaves locally in a linear manner. The structure of the controller itself is often limited to being linear at any point in time, but the "constants" in the controller can change slowly over time as it adapts. Even with all of these assumptions of linearity, the entire system consisting of both an adaptive controller and a plant is nonlinear while the parameters are adapting. This has made it very difficult to prove that these controllers are stable, although recent progress has been made in this area [Åst83].

Adaptive control systems generally exhibit some delay while they are adjusting, particularly when noisy sensors are used (since filtering creates additional delay). If the characteristics of the plant vary considerably over its operating envelope (e.g., due to nonlinearity), an adaptive controller based on a linearized model of the plant can end up spending a large percentage of its time in a "partially" adapted state, leading to degraded performance. Moreover, the control system will have to readapt every time a new regime of the operating envelope is entered.

2.2.5 Gain Scheduled Control

Although a system with significant nonlinearities could be controlled by an adaptive controller which adjusts to the local linear dynamics in each region of the operating envelope, most control systems handle nonlinearities with gain scheduling. Such controllers are collections of simple proportional controllers, one for each distinct region of the operating envelope. For example, in a typical complex control system, the state vector might include 10 elements, three of which are special. When these three are kept constant, a simple, linear control law can work well. The commands sent to the actuators can be a dot product of the state vector and a gain matrix. When any of the three special elements change though, a new linear control law with new gains must be used. In a gain scheduled

controller, the space of all possible values for those three state vector elements is divided up into distinct regions. Each region employs a different set of gains, and a scheduler is used to smoothly transition whenever the state moves from one region to another. The drawback to this approach is that it usually requires a good model of the plant, as well as large amounts of heuristic manual tuning to decide where the boundaries between regions should be, and what the corresponding control law in each region is. Once the controller is built, it cannot change to accommodate a slowly changing plant, such as a robot where bearings wear and parts age. This control technique does respond instantly, though, when it enters a new region, while the adaptive controller would have to wait for more information before it could adapt to a new region. For these and other reasons (e.g., stability), gain scheduled control is generally used instead of adaptive control in most complex systems today.

2.3 CONNECTIONIST LEARNING CONTROL APPROACHES

A number of different approaches have been suggested for using learning systems in control [Fu86][Bar89]. These systems generally try to solve one of three control problems: produce specified control signals, follow specified trajectories, or optimize specified reinforcement/cost signals. For each of these problems there are one or more different approaches which have been tried, the most common of which are described below.

2.3.1 Producing Specified Control Signals

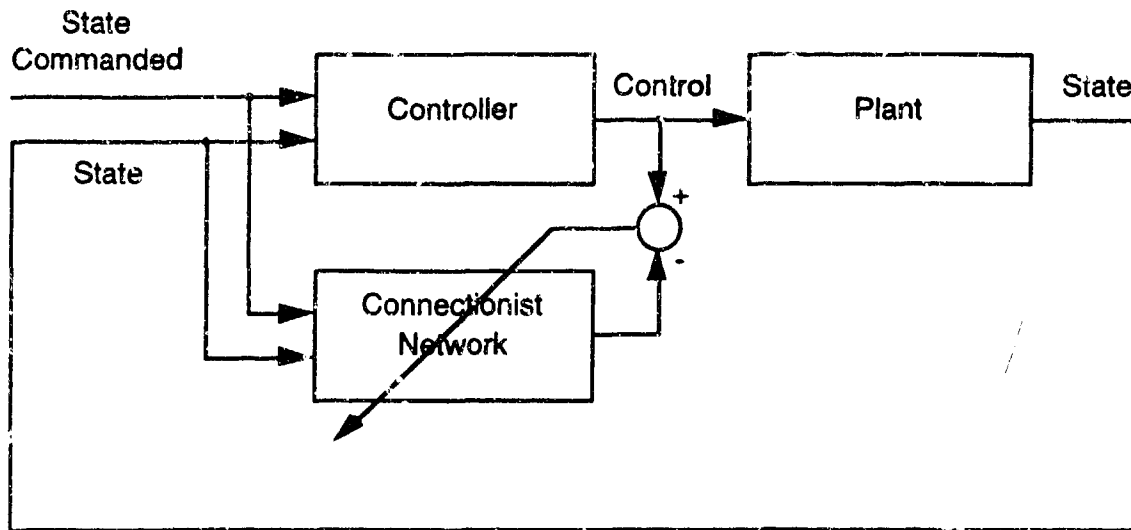


Figure 2.3 Learning specified control signals

The simplest use of a connectionist network in a control application is to emulate an existing controller. This is shown in figure 2.3. The controller and the network are both told the current state and the commanded state (the state to which the controller should drive the plant). The controller then calculates an appropriate control signal by some means, and the network also calculates a control signal. If they differ, the difference is the error in the network's output and is used to train the network (shown by the diagonal line through the network). In the figure shown, the network has no effect on the behavior of the system, it is simply a passive observer. Once the network has learned, the weights in the network would be frozen, and the original controller would be completely removed from the diagram and replaced by the network. One early network, Widrow's ADALINE in the 1960's, was trained to balance a pole on a cart by watching a human do it, and learning from that example [Wid89]. Almost any general supervised learning or automatic function approximation system can be used to control a system in this manner, although the

technique is obviously limited to systems where a control system already exists. This approach might actually be useful in situations where it would be too expensive or too dangerous to have a human controlling a system, but where a network could be used fairly cheaply. It would also be useful if it could be trained by example to control under certain conditions, and then could generalize to others. These are unlikely to be very common uses of such a system.

A more widely applicable use may be as a component of a larger control system that learns to reproduce the results of the other components. For example, a control algorithm may require an extensive tree search on each time step that takes too long to implement in real-time, even in hardware. If it is possible to train a network to implement the same mapping from state to outputs, then the network could replace the slow controller.

2.3.2 Following Specified Trajectories

A much more common control problem is that of following specified trajectories. If the plant being controlled is fairly well understood, and if it is not very nonlinear, then it is often possible to specify a trajectory for the plant which is known to be both useful and achievable. For example, if a robot arm is told to move from its current position to a new position, the ideal behavior might be for it to instantaneously move to that position, and completely stop moving as soon as it reaches it. This, unfortunately, requires the application of infinite force to the arm. On the other hand, it requires very little force to move the arm to the new position quickly but with a large amount of overshoot and oscillation once it gets there, or to move it to the position slowly but with little overshoot. There is a trade-off between force applied, time to get to the correct position, and time to settle once it is there. The exact nature of the trade-off depends on the particular equations governing the arm. Often, through partial models of the plant, trial and error, and experience with similar plants, it is possible for a control engineer to choose a particular trajectory for the arm that is achievable and that gives acceptable behavior for the particular

application.

Choosing the reference trajectory may or may not be difficult in a given situation, but it is extremely important. If the reference trajectory is not very demanding (causing the state to approach the desired state very slowly or allowing a large amount of overshoot), then the system will not perform as well as it could with a better controller. If the reference trajectory is too demanding (causing the state to approach the desired state rapidly with little overshoot), then the controller will attempt to use control actions outside the range of what is possible, and the system may become unstable.

Once such a reference trajectory has been found, then the controller must simply act at each point in time so as to move the plant along that reference trajectory. Three approaches for using connectionist learning systems in "model reference" control problems have been explored: learning a plant inverse, dynamic signs, and Backpropagation through a learned model.

Learning a Plant Inverse

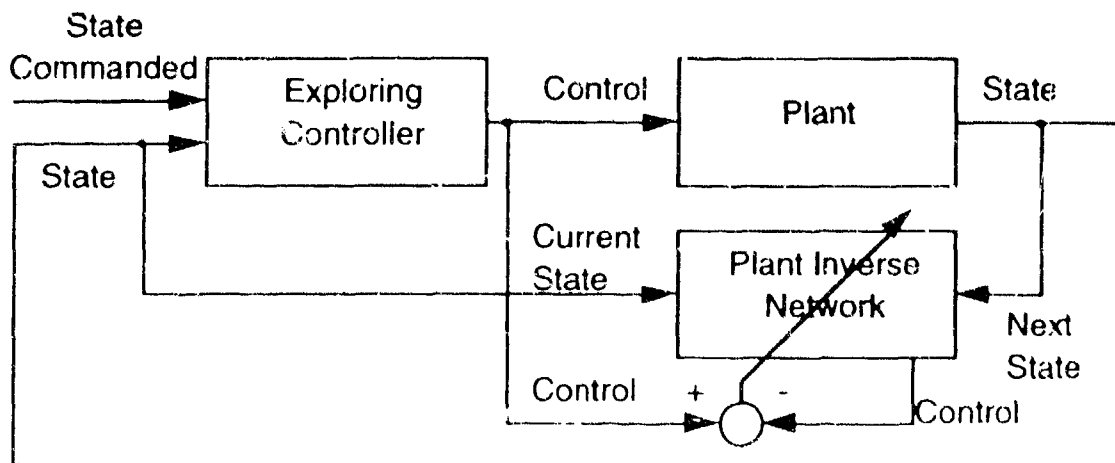


Figure 2.4 Learning a plant inverse

ATTACHMENT 1

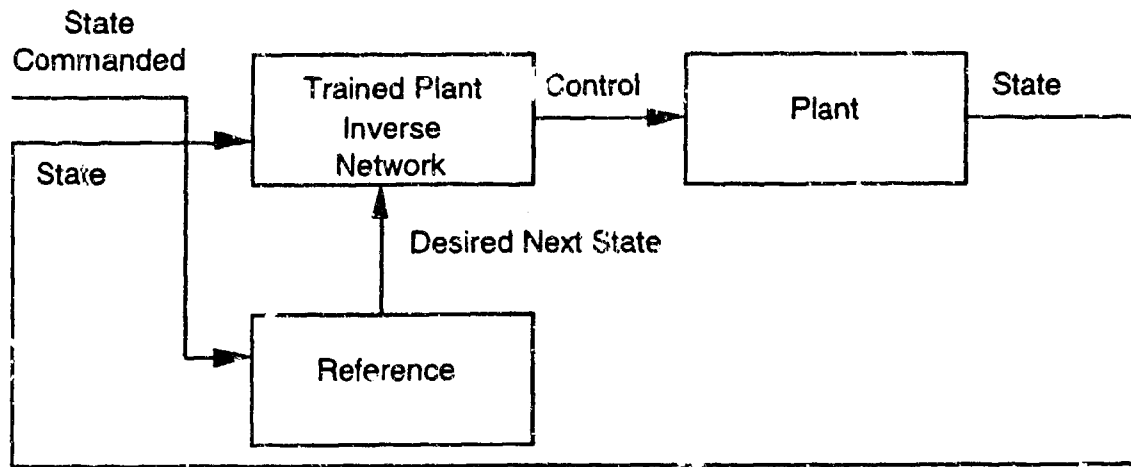


Figure 2.5 Using the plant inverse

A conceptually simple approach to model reference control is to use a learning network to learn a plant inverse. In a deterministic plant, the state of the plant on a given time step is a function of both the state and control action on the previous time step. Alternatively, in continuous time, the rate of change of state at a given point in time is a function of the state and control action at that point in time. An *inverse* of this function with respect to the control signal is a useful function to know. Given the current state and the desired next state (or desired rate of change of state), an inverse gives the control action required. If a network can learn such an inverse, then it can calculate the control actions on each time step that will cause the plant to follow a desired trajectory.

Figure 2.4 illustrates how a network is trained to learn the plant inverse. First, some kind of exploring controller is used to drive the plant. This may not be a very good controller; in fact it could even behave randomly. Its purpose is simply to exercise the plant and show examples of various actions being performed in various states. The network then takes two inputs: the plant's state at the current time and the plant's state on the previous time step. The output of the network is then its estimate of the control action that caused the plant to make the transition from one state to the other. This estimate is then compared to the actual command to generate the error signal used to train the network.

Figure 2.5 shows how the network is used after it has learned. Given the current

ATTACHMENT 1

state of the plant and the desired next state (as specified by the reference trajectory), the network generates a control action to move the plant to that new state. If the next state of the plant does not match the desired state perfectly, then this error could be used to continue training the network. In this way, the network could learn to control a plant whose dynamics gradually change over a long period of time.

A fundamental problem with learning the inverse of the plant is the network's behavior when the plant does not have a unique inverse. Most network architectures, when trained to give two different outputs for the same input, will respond by learning to give an output that is the average of the training values. For example, if a plant at a particular state can be forced to act in the desired way by giving a control signal of either 1 or 3, the network will usually learn an output of 2 for that state, which may be a far worse action than either 1 or 3.

If the plant is a stochastic system, then the result of a single action will be an entire probability distribution function, which further complicates the problem of learning either the forward or inverse model, and of choosing the best action. These problems often limit the usefulness of learning plant inverses.

Dynamic Signs

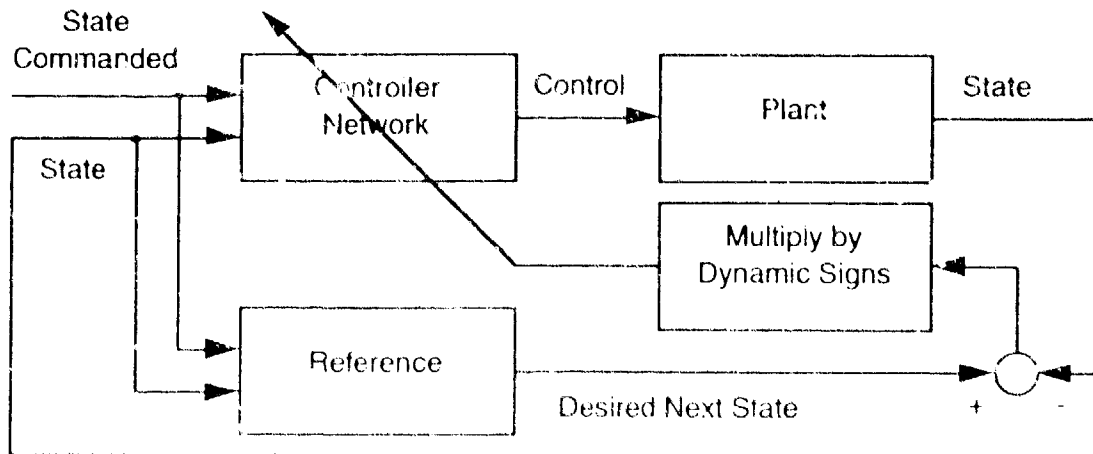


Figure 2.6 Learning with Dynamic Signs

ATTACHMENT 1

A learning system using dynamic signs is shown in figure 2.6. For a given state, the network tries to find an action that will drive the plant to the next state on the trajectory defined by the reference. If it does this, then the new state will equal the output of the reference, and the subtraction will yield zero error, so no learning will occur. On the other hand, if there is an error in the state, then each weight in the network should be adjusted proportionally to its effect on that error. Finding the effect of a given weight on the control signal is easy; it is simply the partial derivative of the control with respect to that weight. To find its effect on the plant's state, however, it is necessary to know the partial derivative of the state of the plant with respect to the control signal.

Often the general behavior of a plant is known, even though all the exact equations and constants are not known. For example, it is often clear that applying more control action will cause one element of the state to increase and another one to decrease, even though it is not possible to predict exactly how much change will occur. In this case, the partial derivative of state with respect to control is not known, but the sign of the partial derivative is known. If the actual partial derivatives were known, then the error in state would be multiplied by the derivative before being used to train the network. Since only the sign of the derivative is assumed known, each element of the error is merely multiplied by plus or minus one. Figure 2.6 shows how the error in the state is multiplied by this value before being used to train the network. This "dynamic sign" has been shown in some cases to contain enough information to cause the network to converge on a reasonable controller [FGG90]. It has been shown [GF90][BF90] that for autonomous submarine control with a multidimensional state vector and a scalar control, the system can learn to be an effective controller using dynamic signs.

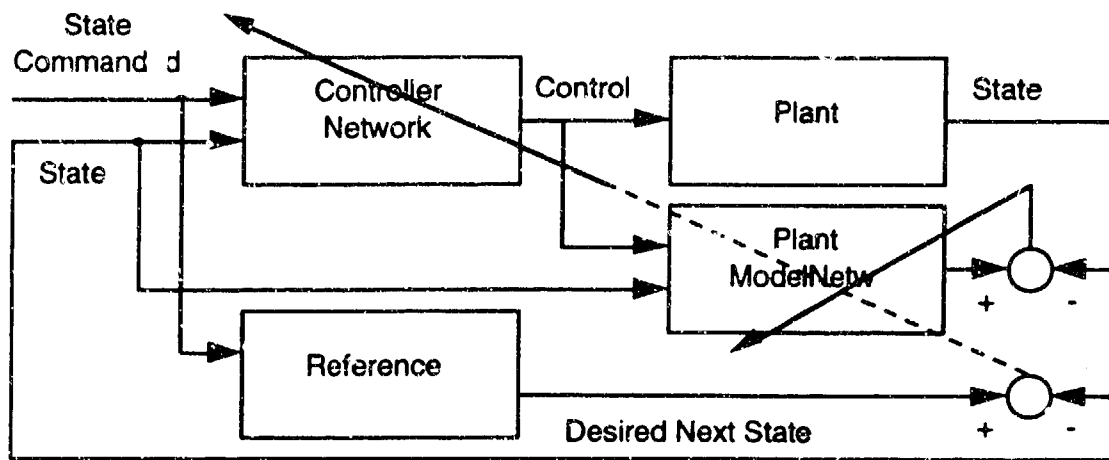
Backpropagation Through a Plant Model

Figure 2.7 Backpropagation through a plant model

A more general approach than dynamic signs is for one network to act as a controller while a second network learns to model the plant. On each time step, the second network takes the current state and control actions as input, and tries to predict what the change in state will be, adjusting its parameters according to the error in its prediction. If the second network is differentiable everywhere, which is the case in networks that use Backpropagation, then when it learns the model, it will also know all of the partial derivatives for the plant. This then allows errors in the state to be backpropagated through both networks in order to change the parameters of the first network so that it can learn to control the plant model. This is the same as the dynamic signs approach described above, except that the partial derivatives across the plant are estimated automatically instead of being set to plus or minus one by hand according to *a priori* information.

Figure 2.7 illustrates this process. The network on the right is trained to predict what the next state of the plant will be, given the current state and control. This training is indicated by the solid diagonal arrow through the network. At the same time, the network on the left is trained to be a better controller. This is done by propagating the error in state through both networks, while only changing weights in the controller network. Although

this signal propagates through the plant model network, it is not used to train that network, which is why it is represented in the diagram by a dotted arrow. This approach has been successfully used by Jordan [Jor88].

2.3.3 Optimizing Specified Reinforcement Signals

The above techniques are all based on the assumption that there is a reference trajectory to follow. At each time step, given the current state, it is assumed that the desired change in state is known. For some systems though, finding a reference trajectory is fully as difficult as finding the controller in the first place. For example, a large semi truck consists of two sections with a hinge between them. If the truck is near a loading dock and at an angle to it, it can be difficult to calculate how to back up the truck so that it ends up with the back end lined up with the dock [NW89]. This procedure may involve turning the wheel all the way to the left, backing up some, then gradually turning it to the right, then finally straightening it out, causing the truck to follow an "S" shaped path. If the path to follow is known, it is trivial to calculate how to turn the wheel to follow the path, but finding the correct path in the first place is a difficult problem. The model reference systems discussed above are therefore not useful for solving this type of problem. In this case, the goal is actually to minimize a quantity after a certain period of time (the distance from the dock at the end), rather than to follow a given trajectory.

This is just one example of the most general type of control problem, which is the optimization of some quantity over time. This is called "Reinforcement Learning" since the goal of the controller is to maximize some external reinforcement signal over time [Wil88]. Since several actions may be performed before the reinforcement is received, it is often difficult to determine which of the actions were good and which were bad. This "temporal credit assignment problem" makes reinforcement learning the most difficult type of problem considered here. Control problems of this type include backing up a truck to minimize the error at the end, finding the route to the moon that requires the least fuel, or finding the

actions for an animal that maximize the amount of food it finds. All of these cases involve maximizing a reinforcement (or minimizing a cost) over some period of time (finite or infinite). This is a difficult problem, since it may be necessary to perform actions that appear "worse" in the short run, but are "better" in the long run. If a controller generates some action and then receives negative reinforcement (or positive cost), it is not clear whether that is the immediate result of that action or the delayed result of a much earlier action. Thus it is not clear how to learn the correct action, or even how to evaluate a given action.

This difficult control problem has been addressed by Backpropagation through time, actor-critic systems, and dynamic programming systems. Actor-critic systems and dynamic programming systems tend to be broad categories with some overlap, but are a useful way of classifying the many approaches to this type of problem.

Backpropagation Through Time

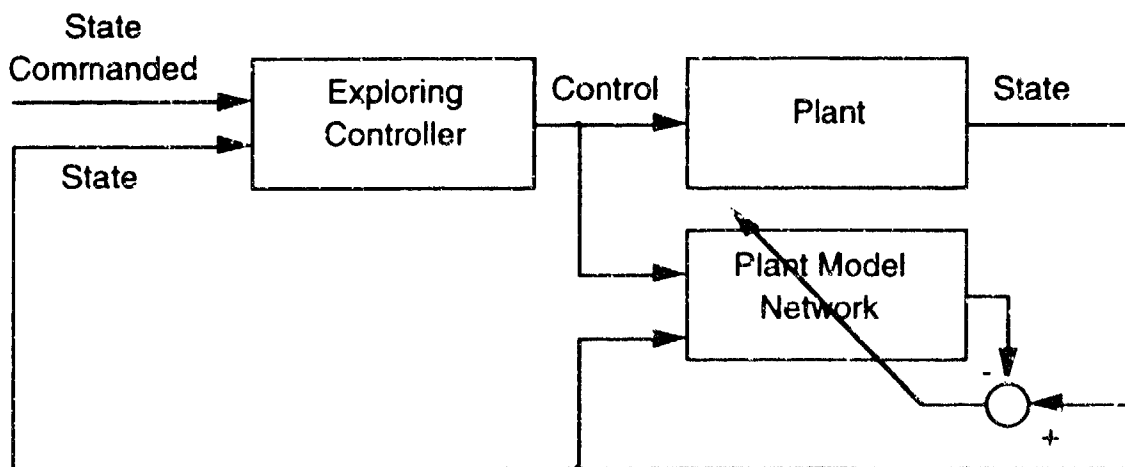


Figure 2.8 Backpropagation through time: learning the plant model

One way to solve the reinforcement learning control problem is to extend the idea of backpropagating through a plant model. Two networks are used. One is trained every time

ATTACHMENT 1

step to learn to model the plant. Figure 2.8 shows how the network can learn to model the plant using the current state, the previous state, and the previous control action.

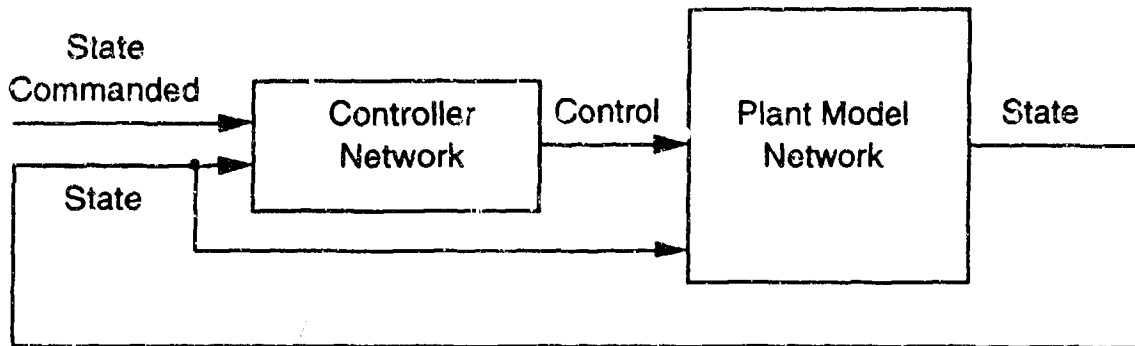


Figure 2.9 Backpropagation through time: learning the controller

Once it has learned, the second network can learn to be a controller based on the plant model. The two networks are connected as shown in figure 2.9. With all parameters fixed, the plant model network starts at some initial position, and the controller network controls the model for a period of time that is known to be long enough to drive the plant to the desired state. All of the signals going through the networks are recorded during the trial.

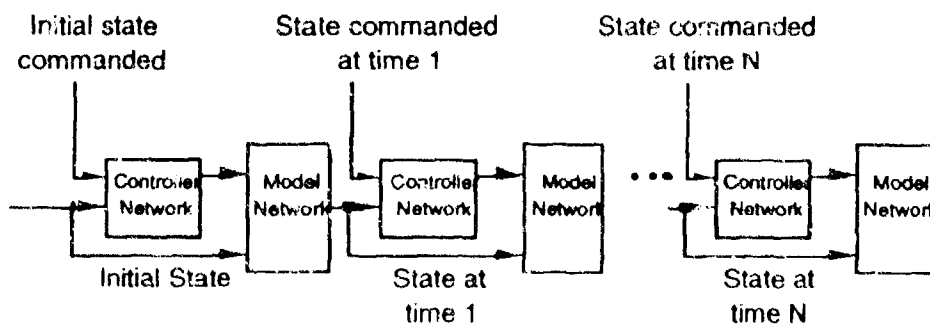


Figure 2.10 The networks unrolled in time

The two networks are then "unrolled in time", so that it looks like the signals have

passed through a very long network once, instead of passing through two small networks many times. The cost or reinforcement signals are calculated from the plant model state at certain time steps of the "unrolled" network. In the case of the truck backer-upper example [NW89], this signal is zero on every time step until the end, and then is equal to the error in state after the last time step. This error can be backpropagated through the large "unrolled" network to change all of its parameters, thus forcing the controller to be slightly better throughout the whole trial. This "Backpropagation through time" procedure has been shown to be able to solve the problem of backing up a truck [NW89]. It is related to ideas suggested by Werbos [Wer89] and to work done by Jordan [Jor88] and Jarneson [Jam90] where signals are propagated back through time during training.

Backpropagation through time does have the difficulty, unfortunately, of requiring that every signal on every time step during each trial be saved. For long trials this could be a problem. Other algorithms could be used instead, such as the Williams-Zipser algorithm for training recurrent networks [WZ89]. This has memory and processing requirements that are independent of the length of the trial, but proportional to the cube of the number of nodes (assuming fully interconnected nodes), so that it can also be impractical for large networks.

Actor-Critic Systems

Backpropagation through time is potentially a very useful technique, but is still not completely general. Even assuming the networks can perfectly model the functions they are trained with, the result will still be a controller that causes the plant to follow a locally optimal path. The path will be such that any small change to it will make it worse, although a large change to the path as a whole might still improve it significantly. The Backpropagation through time algorithm also requires storing all of the signals going through the network throughout the whole trial. In a regulator problem, where the the plant may never fail and may never reach the goal state exactly, the trial will be infinitely long.

An alternative approach that avoids some of these difficulties is to use a system with two components, called an "actor" and a "critic." The actor is the actual controller that, given the state, decides which control actions should be used. The critic is a component that receives external reinforcement signals and uses them to train the actor. This is still a difficult problem, since reinforcement may come long after the actions that caused it. In fact, the best actions may actually increase errors before they start to decrease them, and the critic must recognize that this is the case. For example, with the cart-pole system, if the cart starts at the origin with the pole balanced, and the goal is to move one meter to the right, the reinforcement on each time step might be the negative of the position error. The fastest way to move the system one meter to the right without allowing the pole to fall over, is to first move left, causing the pole to tilt to the right, and then move quickly to the right. Thus the error in position should increase before it decreases. If the actor is to learn the control actions that will accomplish this, the critic must first learn to recognize that this is desirable. It will have to learn that a large position error with the pole tilted the correct way is sometimes preferable to a smaller position error with the pole tilted the wrong way.

Samuel's checker player [Sam59] was one of the earliest systems to take this approach. The actor was an algorithm that switched between book playing and an alpha-beta tree search. The search was based on the relative desirability of various board positions, as determined by the critic. The critic was a linear combination of several hand-built heuristic functions, and learning for the critic consisted of adjusting the weights of the linear combination, and also deciding which of a large number of heuristic functions should be included in the combination.

Michie and Chambers [MC68] developed the Boxes system which consisted of an actor and a simple critic. They applied their controller to a cart-pole system that would signal a failure whenever the pole fell over. The critic based its evaluation of a particular state on the number of time steps between entering that state and failure. This system was later improved by Barto, Sutton, and Anderson [BSA83] with the development of the

Associative Search Element (ASE) and Adaptive Critic Element (ACE). In that system, the critic based its evaluation on both the time until failure and the change in evaluation over time. Evaluations were therefore both predictions of the desirability of a given state, and estimates of what the evaluations would be in future states. This system learned to balance a pole on a cart more quickly than the Boxes system.

Dynamic Programming Systems

Dynamic programming is a class of mathematical techniques for solving optimization problems. Often in a problem the sets of possible states and actions are finite, or can be approximated as finite sets. The problem is to find the best control action for each state, taking into account that it may be profitable to perform actions with low reinforcement (or high cost) in one state in order to reach another state that gives high reinforcement (or low cost). Not only is an action associated with each state, but typically one or more other values are associated with each state as well.

The most common formulation of dynamic programming associates two values with each state. A "policy" is the action that is currently considered to be the best for a given state. An "evaluation" of a state is an estimate of the long-term reinforcement or cost that will be experienced if the system starts in that state and performs *optimal* actions thereafter. All policies and evaluations are initialized to some set of values, and then individual values are improved in some order. A given policy or evaluation is improved by setting it equal to the value that would be appropriate for it if the values of its neighbors were correct. If this process is done repeatedly to policies and evaluations in all the regions, then under certain circumstances it is guaranteed to converge to the optimal solution [WB90]. The set of policies function somewhat as an actor, while the set of evaluations function as a critic. Reinforcement learning with actor-critic systems may therefore sometimes be thought of as a kind of dynamic programming.

Other types of dynamic programming systems do not resemble actor-critic systems.

ATTACHMENT 1

Q learning, devised by Watkins [Wat89], only involves one type of value. For each possible action in each possible state, a number (the "Q value") is stored that represents the expected long-term results if that action is performed in that state followed by optimal actions thereafter. As in the other forms of dynamic programming, a Q value is updated by changing it to be closer to the value that would be appropriate for it if the Q values of all its neighbors are assumed to be correct. Q learning is also guaranteed under certain assumptions to converge to the optimal solution.

The above discussion assumed that the sets of possible states and actions were finite. If there is a continuum of states and actions, then an approximation to dynamic programming must be used. The most common approximation is to divide the state-space into small regions, and store evaluation and policy values for each region. If the state-space is high-dimensional, this will require prohibitively many values to be stored, and dynamic programming is not feasible. A natural solution to this "curse of dimensionality" is to use some form of function approximation system to store the evaluation and policy for the continuous set of states. Connectionist systems are a natural candidate for this use.

This section has described systems for solving the problems of emulating a specific controller, following a specified trajectory, and optimizing a specified signal. None of the systems described here make use of much *a priori* knowledge of the plant. Often, fairly accurate models of a plant exist, and it would be useful to have some method for quickly integrating this knowledge into the controller. The systems described here also tend to react very slowly to changes in the plant, since the network must learn a new function whenever the plant changes. These are problems that this thesis addresses.

3 HYBRID CONTROL ARCHITECTURE

The architecture presented here represents a new method of integrating *learning* and *adaptation* in a synergistic arrangement, forming a single hybrid control system. The adaptive portion of the controller provides real-time adaptation to time-varying dynamics and disturbances, and initially accommodates any unknown dynamics. The learning portion deals with static or very slowly changing spatial dependencies. The latter includes any aspect of the plant dynamics that varies predictably with the current state of the plant or the control action applied.

A conventional adaptive control system reacts to discrepancies between the desired and observed behaviors of the plant to achieve a desired closed-loop system performance. These discrepancies may arise from time-varying dynamics, disturbances, sensor noise, or unmodeled dynamics. The problem of sensor noise is usually addressed with filters, while adaptive control itself is used to handle the remaining sources of observed discrepancies. In practice, little can be done in advance for time-varying dynamics and disturbances; the control system must simply wait for these to occur and then react. On the other hand, unmodeled dynamics that are purely functions of state can be *predicted* from previous experience. This is the task given the learning system. Initially, all unmodeled dynamics are handled by the adaptive system; eventually, however, the learning system is able to *anticipate* previously experienced unmodeled dynamics.¹ Thus, the adaptive system is free to react to time-varying dynamics and disturbances, and is not burdened with the task of reacting to predictable, yet initially unmodeled dynamics.

The hybrid adaptive / learning system presented in this thesis accommodates both

¹This assumes, of course, that the order of the plant (dimension of its state vector) is accurately known.

ATTACHMENT 1

temporal and spatial modeling uncertainties. The adaptive part has a temporal emphasis; its objective is to maintain the desired closed-loop behavior in the face of disturbances and dynamics that are time-varying or appear to be time-varying (e.g., a change in behavior due to a change in operating conditions). The learning part has a spatial emphasis; its objective is to facilitate the development of the desired closed-loop behavior in the presence of unmodeled nonlinearities within the operating envelope. Typically, the adaptive part has relatively fast dynamics, while the learning part has relatively slow dynamics. The hybrid approach allows each mechanism to focus on the part of the overall control problem for which it is best suited, as summarized in Table 3.1.

| ADAPTATION | LEARNING |
|---|---|
| reactive: maintain desired closed-loop behavior | constructional: synthesize desired closed-loop behavior |
| temporal emphasis | spatial emphasis |
| no memory \Rightarrow no anticipation | memory \Rightarrow anticipation |
| fast dynamics | slow dynamics |
| local optimization | global optimization |
| real-time adaptation (time-varying dynamics) | design & on-line tuning (spatial dependencies) |

Table 3.1. Adaptation vs. learning.

A schematic of one possible realization of a hybrid adaptive / learning control system is shown in Figure 3.1.

ATTACHMENT 1

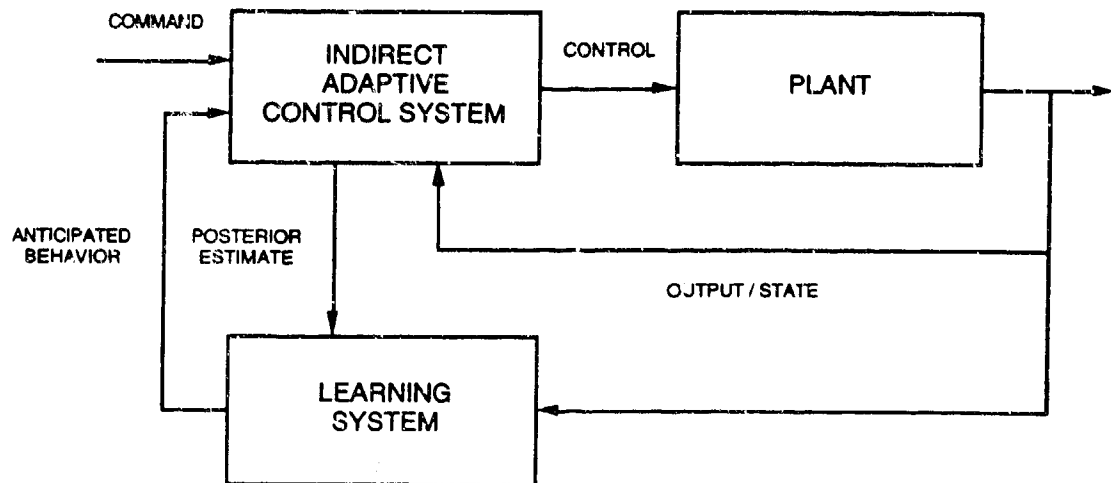


Figure 3.1 Hybrid adaptive / learning controller

To simplify the discussion, we assume that all necessary plant state variables are observable and measured; in the event that this is not the case, a state observer would have to be used. The indirect adaptive controller outputs a control action based upon the current state, the desired state, and the estimated behavior of the system being controlled. This estimate characterizes the current dynamical behavior of the plant. If the behavior of the plant changes, the estimator within the adaptive controller will update the model. If plant changes are unpredictable, then the estimator will attempt to update the model as quickly as possible, based on the information available in the (possibly noisy) sensor readings. Adapting to predictable model errors that are functions of state will take just as long as adapting to unpredictable disturbances and temporal changes, assuming similar noise levels.

The problem of accommodating predictable spatial dependencies is handled by the learning system in the outer loop. It monitors the indirect adaptive controller's posterior estimate of the plant parameters, and learns to associate the appropriate plant parameters with each point in the state-space. The learning system can then anticipate plant behavior based on past experience, and give its prediction to the indirect adaptive controller. This allows the controller to accommodate predictable dynamics while still retaining the

capability to rapidly adapt to unpredictable dynamical effects.

The learning system used here is a feedforward multilayer network. The input is the current state, and the output is a *prediction* of what the plant parameters should be, given that state. Prior to learning, the network is initialized so that the hybrid controller will give the same control signals that the adaptive controller would give by itself. When the network has correctly learned the mapping, the hybrid adaptive / learning controller will anticipate nonlinear model errors that are functions of state and are predictable, and will respond faster and more efficiently than a simple adaptive controller would. If these spatial dependencies change, then the hybrid will act as an adaptive controller until it learns the new mapping. The entire system is automatic; no explicit switching mechanism is needed to go from adaptation to learning.

Note that any type of connectionist network could be used for the learning system, it need only have the ability to learn functions from examples. This part of the system could even be some other form of associative memory such as a lookup table or a nearest neighbor classifier. In practice, though, these types of techniques may be impractical since a potentially infinite number of example points are used for training, and the state-space may have a high number of dimensions. For this reason, a connectionist network seems more appropriate.

3.1 THE LEARNING COMPONENT

The hybrid architecture allows any learning system to be used that can learn to approximate a function from a large set of examples of that function. The first learning system examined here was a feedforward, Backpropagation, sigmoid network. The inputs to the network and the outputs from the network were scaled to vary over a range of unit width. The training examples were stored in a large buffer, and were presented to the network in a random order. The network was trained incrementally; weights were changed

after each training example was presented.

The network was then tried with a different learning algorithm, a heuristic method that approximates the effect of using the Hessian to scale the weight changes. A modified version of this algorithm was also tried.

The learning systems, and the reasons behind their choice for this application, are described in further detail in chapter 4.

3.2 THE ADAPTIVE COMPONENT

The adaptive component of a hybrid controller can be any indirect adaptive controller that can incorporate outside information. The controller might, for example, estimate parameters of the plant, and then act as the best controller for those parameters. It might instead estimate the amount of error in its predictions of the new state on the next step, and try to compensate for it. For the experiments performed here, an indirect adaptive controller of the latter type was used, both in its original form and with modifications.

A technique based on Time Delay Control (TDC) was chosen as the adaptive system for the experiments presented here. TDC is an indirect adaptive control method developed by Youcef-Toumi and Osamu [YI90].

This system works by looking at the difference between the current state of the plant and the state of the plant on the previous time step. This difference, along with knowledge of what action was chosen on the previous time step, is used to estimate the effect that the unmodeled dynamics are having on the system. This value \hat{h} is calculated explicitly and plays a pivotal role in the remaining calculations. A control action is then generated to cancel the unwanted effects (modeled and estimated) and to induce the desired behavior in the plant. The technique uses information that is only one time step old, so it is able to react to sudden changes in the plant or environment after a single time step. Of course, since it is in effect differentiating the state, it is sensitive to high frequency noise.

Youcef-Toumi points out that this is not as bad as it seems if the plant itself acts as a low-pass filter, attenuating the effect of the noise in the control actions.

The controller can also be made less sensitive to high frequency noise by simply using a larger time step and a filter. This, however causes it to react more slowly to changes in the plant. Overall, TDC does a good job, but it cannot both react quickly and remain insensitive to high frequency noise.

3.3 THE HYBRID SYSTEM

The connectionist network used in the hybrid adaptive / learning controller is a simple, feedforward, back-propagation network, with two hidden layers of ten nodes each. Given the state and goal for the plant, the network could be trained to output an estimate of the unmodeled dynamics h . In the absence of noise, this should be the same h that TDC calculates. If noise is present, it may be possible to determine the current state of the plant to within a small error. The correct h , however, is difficult to calculate precisely, because it is found by "differentiating" the state (e.g., using a backwards difference).

One property of connectionist networks is useful here. During training, a network is given input and desired output values repeatedly. If it is given conflicting desired outputs for the same input, then it tends to average them. This means that the network can be trained with data that has small, zero mean noise and still learn the correct mapping. Therefore, if TDC calculates noisy h 's with an equal probability of the value being too high or too low for a given state, and if these are used to train the network, then the network will tend to learn the correct h for each state.

Moreover, a learning system is not only useful when h is noisy; it is also helpful when it is used to *predict* h . In its original form, TDC looks at the state of the plant before and after a given time step. Back-differencing to estimate the derivative, TDC can then calculate the unmodeled dynamics h during that period. That h is then used to calculate the

appropriate control action to be applied to the plant during the *next* time step. This is a source of error in the controller, since it is always sending out control actions based on what was correct on the previous time period. With the network, there is a simple solution to this problem. Instead of associating h with the current state during training, it is associated with the previous state. After the network has been trained with those patterns, it should be able to *predict*, given a state, what h will be during the time step following that state. This allows a better estimate to be calculated.

The hybrid controller, therefore, has at least the potential to solve both of the difficulties associated with the original adaptive controller. This is in addition to the main problem it was designed to solve: learning control. These considerations provide motivation for experimenting with the hybrid controller.

The hybrid adaptive / learning controller typically runs at a speed such that the states and h do not change much over a period of several time-steps. If the network is trained on similar states several times in a row, it may "forget" what it knows about other states. One solution might be to train the network less frequently, such as once a second. This might be effective, but it would slow down learning by not learning every time step. A better solution is to use a *random buffer*. During training, as the plant wanders through the state-space, the data from each time step is stored in the buffer. One point is also chosen at random from the buffer on each time step, and is used to train the network. This ensures that the network is trained on a distributed set of points.

3.4 DERIVATION OF THE HYBRID WITH KNOWN CONTROL EFFECT

The original TDC equations were designed to allow the incorporation of *a priori* information consisting of a linear model of the plant. The effect of control action on state was assumed to be known perfectly, but the other parameters could initially be incorrect. The following is a derivation of the TDC equations for a discrete time plant where the

ATTACHMENT 1

known dynamics are described by the *a priori* matrices Φ and Γ , as well as the knowledge gained by the learning system Ψ . As in the original TDC, the effect of control on state is assumed to be a linear function, and the constant Γ is assumed to be known without any error.

Assume that the plant being controlled is of the form

$$\mathbf{x}(k+1) = \Phi\mathbf{x}(k) + \Gamma\mathbf{u}(k) + \Psi(\mathbf{x}(k)) + \mathbf{h}(\mathbf{x}(k), k) \quad (1)$$

where at time k , \mathbf{x} is the state vector, \mathbf{u} is the control vector, and all of the unknown dynamics are represented by the function \mathbf{h} . The vector Ψ is the output of the learning system.

The reference system has the dynamics

$$\mathbf{x}_m(k+1) = \Phi_m\mathbf{x}_m(k) + \Gamma_m\mathbf{r}(k) \quad (2)$$

where \mathbf{r} is the command vector. The error between the actual state and the reference state is

$$\mathbf{e}(k) = \mathbf{x}_m(k) - \mathbf{x}(k) \quad (3)$$

The goal is to build a controller that will cause the error to behave as:

$$\mathbf{e}(k+1) = \{\Phi_m + \mathbf{K}\}\mathbf{e}(k) \quad (4)$$

where \mathbf{K} is the error feedback matrix which allows the error dynamics to be specified independently of the values of the other parameters.

Substituting (3) into the left side of (4), then substituting (2) into the result and solving for $\mathbf{x}(k+1)$ gives the desired next state:

ATTACHMENT 1

$$\mathbf{x}_m(k+1) - \mathbf{x}(k+1) = \{\Phi_m + \mathbf{K}\}\mathbf{e}(k)$$

$$\Phi_m \mathbf{x}_m(k) + \Gamma_m \mathbf{r}(k) - \mathbf{x}(k+1) = \{\Phi_m + \mathbf{K}\}\mathbf{e}(k)$$

$$\mathbf{x}(k+1) = \Phi_m \mathbf{x}_m(k) + \Gamma_m \mathbf{r}(k) - \{\Phi_m + \mathbf{K}\}\mathbf{e}(k) \quad (5)$$

Setting (1) and (5) equal and solving for \mathbf{u} gives the control law that should be followed in order to achieve the desired next state.

$$\Phi \mathbf{x}(k) + \Gamma \mathbf{u}(k) + \Psi(\mathbf{x}(k)) + \mathbf{h}(\mathbf{x}(k), k) = \Phi_m \mathbf{x}_m(k) + \Gamma_m \mathbf{r}(k) - \{\Phi_m + \mathbf{K}\}\mathbf{e}(k) \quad (6)$$

$$\mathbf{u}(k) = \Gamma^+ \{ \Phi_m \mathbf{x}_m(k) + \Gamma_m \mathbf{r}(k) - \{\Phi_m + \mathbf{K}\}\mathbf{e}(k) - \Phi \mathbf{x}(k) - \Psi(\mathbf{x}(k)) - \mathbf{h}(\mathbf{x}(k), k) \} \quad (7)$$

where, for a matrix \mathbf{M} , $\mathbf{M}^+ = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T$ is the pseudo-inverse of \mathbf{M} . The only unknown in (7) is \mathbf{h} . If \mathbf{h} changes slowly, then it can be approximated by its previous value. Solving (1) for \mathbf{h} and then applying this approximation yields:

$$\mathbf{h}(\mathbf{x}(k), k) = \mathbf{x}(k+1) - \Phi \mathbf{x}(k) - \Gamma \mathbf{u}(k) - \Psi(\mathbf{x}(k)) \quad (8)$$

$$\mathbf{h}(\mathbf{x}(k), k) \approx \mathbf{x}(k) - \Phi \mathbf{x}(k-1) - \Gamma \mathbf{u}(k-1) - \Psi(\mathbf{x}(k-1)) \quad (9)$$

Substituting the approximation (9) into equation (7) gives the final control law

$$\begin{aligned} \mathbf{u}(k) = \Gamma^+ \{ & \Phi_m \mathbf{x}_m(k) + \Gamma_m \mathbf{r}(k) - \mathbf{K} \mathbf{e}(k) - \Phi \mathbf{x}(k) - \Psi(\mathbf{x}(k)) \\ & - \mathbf{x}(k) + \Phi \mathbf{x}(k-1) + \Gamma \mathbf{u}(k-1) + \Psi(\mathbf{x}(k-1)) \} \end{aligned} \quad (10)$$

The controller will adapt to a sudden change in the plant dynamics within one time step. If the time step is short, the controller will respond faster, but will also be more sensitive to noise.

3.5 DERIVATION OF THE HYBRID WITH UNKNOWN CONTROL EFFECT

It is often the case that the exact effect of control action on state is only partially known, just as the dynamics of the state are only partially known. If a learning system can learn the unmodeled dynamics, then the partial derivative of the learned function Ψ with respect to control action u will represent the unmodeled effect of control on state, and can be used to improve the *a priori* estimate of this value Γ . The following is a derivation of the hybrid system, incorporating these partial derivatives as an improvement over the approach in section 3.4.

Assume that a plant has the following dynamics:

$$x(k+1) = \Phi x(k) + \Gamma u(k) + \Psi(x(k), u(k)) + h(x(k), u(k), k) \quad (11)$$

where the vector $x(k)$ is the state at time k , the vector u is the control, the matrices Φ and Γ and the function Ψ are the *a priori* known and learned dynamics, and the function h represents all of the unknowns, including unmodeled dynamics, nonlinearities as a function of state or control action, and time-varying disturbances.

Once again, the reference system has the dynamics

$$x_m(k+1) = \Phi_m x_m(k) + \Gamma_m r(k) \quad (12)$$

where r is the command vector giving the state to which the plant should be driven. The error between the actual state and the reference state is

$$e(k) = x_m(k) - x(k) \quad (13)$$

and the goal is to build a controller that will cause the error to decrease according to:

$$e(k+1) = (\Phi_m + K) e(k) \quad (14)$$

Substituting (13) into the left side of (14), substituting (12) into the result of that,

ATTACHMENT 1

and then solving for $x(k+1)$ gives the desired state on the next time step.

$$x(k+1) = \Phi_m x_m(k) + \Gamma_m r(k) - (\Phi_m + K)e(k) \quad (15)$$

All known dynamics not defined by Φ and Γ are represented by the function Ψ . This can be learned or stored in any manner that allows the calculation of the partial derivatives with respect to u . When calculating the u for a given time step, it will be necessary to take into account the fact that Ψ may affect the next state differently according to which u is chosen. Figure 3.2 illustrates how Ψ can be approximated by evaluating it at the current state and previous control action, then forming a line through that point with the appropriate slope in the u direction. Equation (16) shows this approximation mathematically.

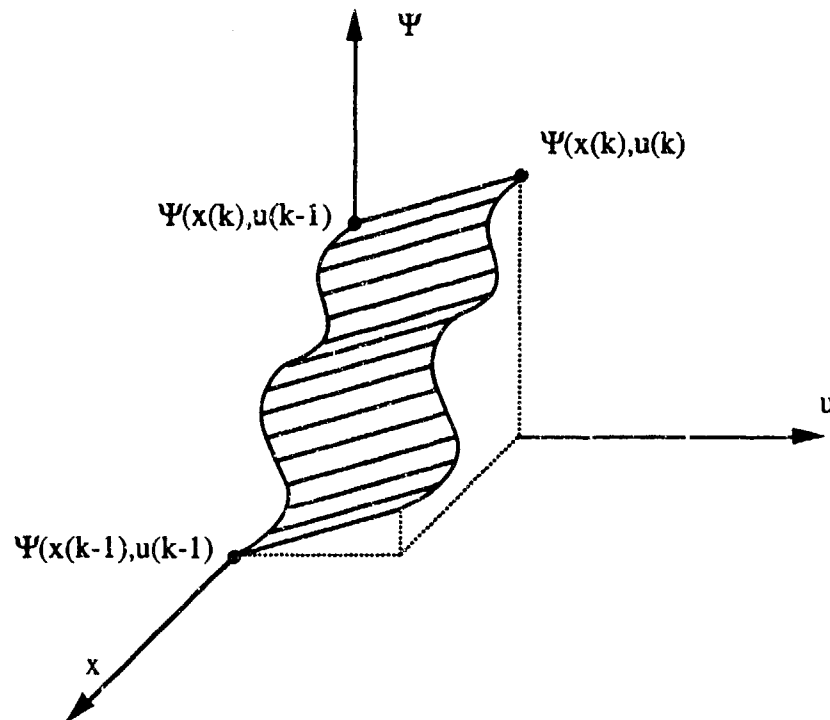


Figure 3.2 Approximation of Ψ as a (linear) function of $u(k)$ and nonlinear function of x

ATTACHMENT 1

$$\Psi(\mathbf{x}(k), \mathbf{u}(k)) \approx \Psi(\mathbf{x}(k), \mathbf{u}(k-1)) + \{\mathbf{u}(k) - \mathbf{u}(k-1)\} \left. \frac{\partial \Psi}{\partial \mathbf{u}} \right|_{\mathbf{x}(k), \mathbf{u}(k-1)} \quad (16)$$

Substituting (16) into (11) gives a more useful formulation of the plant dynamics.

$$\begin{aligned} \mathbf{x}(k+1) = & \Phi \mathbf{x}(k) + \Gamma \mathbf{u}(k) + \Psi(\mathbf{x}(k), \mathbf{u}(k-1)) \\ & + \{\mathbf{u}(k) - \mathbf{u}(k-1)\} \left. \frac{\partial \Psi}{\partial \mathbf{u}} \right|_{\mathbf{x}(k), \mathbf{u}(k-1)} + \mathbf{h}(k, \mathbf{x}(k), \mathbf{u}(k)) \end{aligned} \quad (17)$$

If the function \mathbf{h} representing disturbances, etc. is changing slowly, then it can be approximated by solving for \mathbf{h} in (17) for the previous time step, and using that as the approximation of \mathbf{h} for the current time step:

$$\begin{aligned} \mathbf{h}(k, \mathbf{x}(k), \mathbf{u}(k)) & \approx \mathbf{h}(k-1, \mathbf{x}(k-1), \mathbf{u}(k-1)) \\ \mathbf{h}(k, \mathbf{x}(k), \mathbf{u}(k)) & \approx \mathbf{x}(k) - \Phi \mathbf{x}(k-1) - \Gamma \mathbf{u}(k-1) - \Psi(\mathbf{x}(k-1), \mathbf{u}(k-2)) \\ & - \{\mathbf{u}(k-1) - \mathbf{u}(k-2)\} \left. \frac{\partial \Psi}{\partial \mathbf{u}} \right|_{\mathbf{x}(k-1), \mathbf{u}(k-2)} \end{aligned} \quad (18)$$

Substituting (18) into (17) and solving for $\mathbf{u}(k)$ gives the control law in terms of the desired next state $\mathbf{x}(k+1)$.

$$\begin{aligned} \mathbf{u}(k) = & \left(\Gamma + \left. \frac{\partial \Psi}{\partial \mathbf{u}} \right|_{\mathbf{x}(k), \mathbf{u}(k-1)} \right)^+ \left[\mathbf{u}(k-1) \left. \frac{\partial \Psi}{\partial \mathbf{u}} \right|_{\mathbf{x}(k), \mathbf{u}(k-1)} \right. \\ & - \Phi \mathbf{x}(k) - \Psi(\mathbf{x}(k), \mathbf{u}(k-1)) + \mathbf{x}(k+1) \\ & - \mathbf{x}(k) + \Phi \mathbf{x}(k-1) + \Gamma \mathbf{u}(k-1) + \Psi(\mathbf{x}(k-1), \mathbf{u}(k-2)) \\ & \left. + \{\mathbf{u}(k-1) - \mathbf{u}(k-2)\} \left. \frac{\partial \Psi}{\partial \mathbf{u}} \right|_{\mathbf{x}(k-1), \mathbf{u}(k-2)} \right] \end{aligned} \quad (19)$$

Substituting the desired next state (15) into (19) yields the final control law:

ATTACHMENT 1

$$\begin{aligned}
 u(k) = & \left(\Gamma + \frac{\partial \Psi}{\partial u} \Big|_{x(k), u(k-1)} \right)^+ \left[u(k-1) \frac{\partial \Psi}{\partial u} \Big|_{x(k), u(k-1)} \right. \\
 & - \Phi x(k) - \Psi(x(k), u(k-1)) + \Phi_m x(k) + \Gamma_m r(k) - K e(k) \\
 & - x(k) + \Phi x(k-1) + \Gamma u(k-1) + \Psi(x(k-1), u(k-2)) \\
 & \left. + (u(k-1) - u(k-2)) \frac{\partial \Psi}{\partial u} \Big|_{x(k-1), u(k-2)} \right]
 \end{aligned} \tag{20}$$

4 LEARNING SYSTEMS USED

The learning component of the hybrid control system is responsible for learning the function that the adaptive component discovers *a posteriori*. Because the function is defined over a continuum of states, and can involve a number of dimensions, connectionist systems were chosen for the learning component. First a standard Backpropagation network was used, as described in the next section, then Delta-Bar-Delta learning was tried, as described in the following section, to increase learning speed.

4.1 BACKPROPAGATION NETWORKS

During the operation of an indirect adaptive controller, certain parameters are estimated on each time step, and the controller uses these to choose an appropriate control action. Either on the next time step, or soon thereafter, the controller may have additional information about what the estimates should have been earlier. It is natural to consider whether a learning system of some sort could learn to map the earlier state to later, improved estimates, and so be able to make even better estimates the next time that state is entered. This is simply a delayed function approximation problem.

The function being learned would output parameters as a function of state. The parameters and the state may be high-dimensional vectors, and the function being learned may need to be developed on the basis of a large number of training points generated by the indirect adaptive controller. In this case, a Backpropagation network would seem to be a good model for learning the functions involved. For any given function and desired accuracy, a network can be found that will learn that function to the desired accuracy [HW89]. This is true for networks built from any of a wide range of functions.

ATTACHMENT 1

There are a number of considerations that arise when trying to apply Backpropagation networks to learning functions in this context. First, the data used to train the network comes from a system controlling an actual plant. In this case, the training data consists of states and the appropriate parameters that should be associated with them. The state used for training will always be a recent state of the plant, and since the state of a plant may not change much on each time step, the training data during a given period of time will all tend to come from one region of the state-space. This is even more applicable in the case of a regulator, where the controller tries to keep the plant near a particular state all the time. If the controller is doing a good job and there are no large disturbances, the state of the plant will stay near where it should be. This means that no training points will be generated in other regions. Even in a tracking control problem, the plant may still move slowly through state-space. Therefore, it is important to consider the ability of a given learning system to learn despite repeated exposure to very similar training patterns for long periods of time.

Backpropagation, and most of its variants, all try to adjust the weights in the direction of some gradient and decrease error, as described in chapter 2. The error being minimized J is frequently defined as the mean squared error between the network output and the desired value of its output, summed over all possible inputs:

$$J = \frac{1}{n} \sum_{i=1}^n (f(\mathbf{x}_i, \mathbf{w}) - \mathbf{d}_i)^T (f(\mathbf{x}_i, \mathbf{w}) - \mathbf{d}_i)$$

$$\Delta \mathbf{w}_i = -\alpha \frac{\partial J}{\partial \mathbf{w}_i}$$

where:

J = Total error for network with weight vector \mathbf{w}

n = number of training examples

\mathbf{x}_i = input to network for i th training example

\mathbf{d}_i = desired output of network for i th training example

$f(\mathbf{x}_i, \mathbf{w})$ = actual output of network for i th training example

This implies that the network is be updated by *epoch learning*, where weights are changed

once per *epoch* (once per each pass through all the training examples). However, for the function approximation being done here, the function being learned is continuous. Even if \mathbf{x} is only a two element vector, the error becomes

$$J = \int_{\mathbf{x}_1} \int_{\mathbf{x}_2} (f(\mathbf{x}_i, \mathbf{w}) - d_i)^T (f(\mathbf{x}_i, \mathbf{w}) - d_i) d\mathbf{x}_1 d\mathbf{x}_2$$

This requires summing over an infinite number of training examples, which takes infinite time, just to find the error associated with a single set of weights. The common approximation in this case is to use *incremental learning*. In incremental learning, the weights are adjusted a small amount after each presentation of a training example. The change is made in the direction of the negative gradient of the error associated with only that one example. If the changes are small compared to the time it takes to see all of the inputs, then incremental learning will tend to give the same answer that epoch learning would.

Suppose, however, that increasing a given weight would increase the error for one third of the training examples and decrease it an equal amount for two thirds of the training examples; in this case, the correct action would be to increase that weight. If training examples are presented in a random order, then on each presentation, there will be a one third probability that the weight will decrease and a two thirds probability that it will increase. In the long run, the weight takes a random walk that tends to increase it as it should. If, however, many training points are presented in a row that all have similar inputs and outputs, then their partial derivatives will tend to be similar, and they will all tend to move the weight in the same direction. The net effect of this is to cause the network to learn the function in that region extremely well, at the expense of forgetting any information it had already learned about other regions. This phenomenon is referred to here as *fixation*. One simple method to avoid fixation is to use a buffer to hold many of the training points. Then on each time step a training point can be drawn at random, and used to train the network. This scrambling of the training points helps avoid fixation, but it

ATTACHMENT 1

may require a large memory to hold all of the data.

Another characteristic of Backpropagation is that it tends to learn slowly. There are a number of reasons for this, some of which are clearer when the learning problem is visualized geometrically. The connectionist network contains a finite number of real-valued weights. This weight vector determines the behavior of the network, and so the error is a function of this weight vector. The error can be visualized as a multi-dimensional surface (or manifold) in a space with one more dimension than the number of components of the weight vector. A given weight vector corresponds to a single point on this error surface. The height of the error surface corresponds to the mean squared error associated with that vector. If there is only one training point, there will be an error surface associated with it. If there are several training points, then there is an error surface associated with each of them, and the sum of all those functions gives the total error surface. When a given training point is presented to the network, it is possible to find the partial derivative of the error for that point with respect to each weight. The negative of this gradient corresponds to the direction of steepest descent for the individual error surface associated with that training example. The sum of all the individual gradients gives the gradient for the total error surface.

The goal of learning, then, is to follow the gradient of the total error surface, changing the weights so as to move downhill to a local minimum in that surface. If a certain region of that surface is shaped like a trough, then repeated steps in the direction of the gradient will tend to cause the weight vector to oscillate across the bottom of the trough, and not move very fast in the direction of the gentle slope along the trough. If large steps are taken, then it is possible to leave the trough entirely, perhaps then reaching an undesirable plateau. If small steps are taken, then the weight vector will take reasonable steps across the trough, but will move too slowly along the trough. Such troughs may therefore slow down convergence of gradient descent, and so slow the learning process in a Backpropagation network.

ATTACHMENT 1

Not only do troughs slow down learning, but they are also very common and easily formed. Consider a surface that has a number of roughly circular depressions. If the surface is stretched a hundredfold along one axis, there will then be a large number of troughs parallel to that axis. In the error surface for a network, each weight is one axis. Therefore simply multiplying a weight by a large constant (and backpropagating through that constant appropriately) can create troughs in weight space. Similarly, if one of the inputs to a network varies over a much wider range than another, troughs will tend to form. To avoid this scaling problem, all experiments for this thesis were carried out with all inputs and outputs to and from networks scaled to vary over a range of unit width.

An obvious solution to the problem of troughs would be to look at both the first and second derivative for the current weight vector. Instead of simply calculating the gradient of the error surface at a point, the curvature at that point could also be calculated. Since the gradient changes rapidly across the trough, the curvature in that direction would be large, and small steps in that direction would be appropriate. Since the gradient changes slowly along the trough, the curvature is low in that direction, and it would be safe to take larger steps in that direction. Thus, if the step size in each direction is decreased in proportion to the curvature in that direction, then the modified gradient descent will tend to head more directly towards the local minimum, and can reach it in less time with fewer oscillations. If the trough is actually a very long, thin ellipsoid (i.e., a perfect quadratic function), then dividing by the second derivative would allow the local minimum to be reached in a single step.

Figure 4.1 illustrates a trough with a dot representing the current weight vector. The arrow pointing to the right is the negative gradient, which points mainly across the trough and only slightly along the trough. Taking discrete steps along this gradient can cause oscillation, and could even leave the trough entirely if the steps are too large. The arrow pointing to the left is the negative gradient divided by the curvature of the surface. It points directly toward the local minimum (for this ellipsoidal trough), and is a better path to

ATTACHMENT 1

follow for fast convergence.

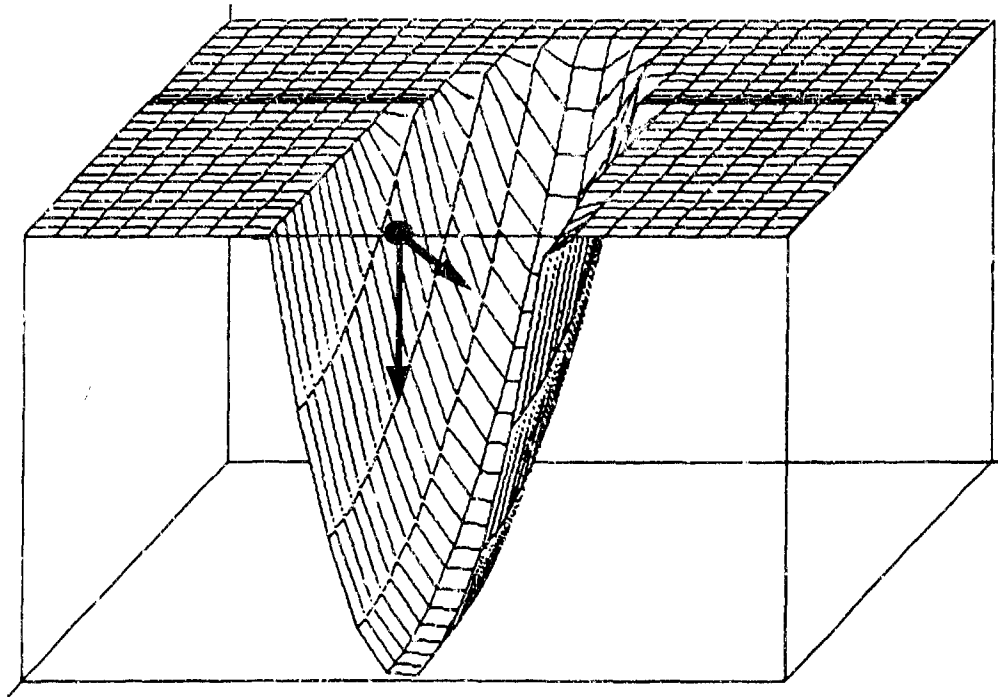


Figure 4.1 A weight vector on the side of a trough, showing the negative gradient (right arrow) and negative gradient divided by curvature (left arrow)

For a multi-dimensional surface, the slope is a vector of first derivatives (the gradient) and the curvature is a matrix of second derivatives (the Hessian). If there are N weights, then the Hessian will be a N by N matrix, and its eigenvectors will point in the directions of maximum curvature. The eigenvalues correspond to the curvature in those directions. If it was useful to *multiply* the step size in a direction by the curvature in that direction, then the gradient could simply be multiplied by the Hessian. Unfortunately, the desired operation is to *divide* the step size by the curvature. This is equivalent to multiplying the gradient by the inverse of the Hessian:

ATTACHMENT 1

$$\Delta \mathbf{w} = -\mathbf{G} \mathbf{H}^{-1}$$

$$\mathbf{G}_i = \frac{\partial J}{\partial \mathbf{w}_i} = \text{gradient}$$

$$\mathbf{H}_{ij} = \frac{\partial^2 J}{\partial \mathbf{w}_i \partial \mathbf{w}_j} = \text{Hessian}$$

$$J = \text{total error}$$

This involves inverting an N by N matrix on each step! This procedure may be computationally expensive, so numerous approximations and heuristics have been proposed to accomplish nearly the same thing.

Computation time is not the only difficulty with using the Hessian. Implementing the above equations requires the calculation of the total error and its derivatives. But for continuous function approximation, these are integrals over an infinite number of points. On each time step, the error, gradient, and curvature can only be calculated for one of these points.

This was also the case when simply following the gradient, but the problem was less severe then. If a small step is repeatedly taken in the direction of the gradient associated with a randomly chosen input, then over time the weight vector will follow a random walk in the direction of the true gradient. This is effective if the steps taken are small, and gradually get smaller over time. Now consider calculating the Hessian on each time step, based only on the derivatives for the current training example. The second derivatives for one example may be small, even if the sum of them over all the examples is large. The weight vector would therefore take large steps when it should be taking small steps.

In the case of a network with only one weight, this problem can be seen algebraically. The correct step size is the total gradient divided by the total curvature. If the steps taken are actually the individual slopes divided by individual curvatures, then the answer is completely wrong:

ATTACHMENT 1

$$\frac{\sum_i s_i}{\sum_i c_i} \neq \sum_i \frac{s_i}{c_i}$$

where:

s_i = slope (first derivative)

c_i = curvature (second derivative)

The left side is correct. The step size should be the total slope divided by the total curvature. The right side is incorrect. It is not useful to look at each individual training point and divide its individual slope by its curvature. In the expression on the left, a small c_i has almost no effect, whereas on the right side it has a very large effect. When learning continuous functions, the summations above are actually integrals over infinite sets of points. If weights are changed after each pass through all the training data, then this problem does not arise. It is only a problem in incremental training where the weights are changed after each individual error is found. When learning functions over continuous input spaces, the Hessian being inverted should actually be the sum of uncountably many Hessians. If it is simply the sum of the last few Hessians instead, then other problems arise since it is representing the curvature at the weight vector from several time steps previous instead of the current weight vector. The more time steps the Hessian averages over (for more accuracy), the greater the danger that it is no longer meaningful. It is not a theoretical certainty that second-order methods such as this are more useful for infinite training sets being trained incrementally, even if the calculations can be done cheaply. Furthermore, the very nature of self-modifying step sizes may make the network more susceptible to fixation if the training points are not picked in a perfectly random manner.

4.2 DELTA-BAR-DELTA

Backpropagation has been modified in a number of ways by different researchers as a means of speeding convergence during learning. These modifications are generally compared with Backpropagation on toy problems with small training sets. The Delta-Bar-Delta algorithm, a heuristic method developed by Jacobs [Jac91], is one such attempt at improving the rate of convergence. It has been shown by Jacobs and confirmed in other work performed at Draper Laboratory that this method sometimes allows faster learning than other more common heuristics, on problems involving small training sets. Testing it on the learning problem here allows a more realistic comparison on a more "real world" problem, involving infinite noisy training sets. One of the goals of this thesis is to determine the applicability of methods such as this to learning systems for control.

Delta-Bar-Delta is a heuristic approximation to the effect of using the main diagonal of the Hessian matrix. This main diagonal contains only the second partial derivatives of the error with respect to each individual weight with respect to itself. Delta-Bar-Delta maintains a local learning rate for each weight, which is heuristic approximation of this second derivative. The equations governing Delta-Bar-Delta [Jac91] for a single weight can be written as:

$$\begin{aligned}
 w(t) &= w(t-1) + \alpha(t) \delta(t) \\
 \delta(t) &= \frac{\partial J(t)}{\partial w(t)} \\
 \bar{\delta}(t) &= (1-\theta) \delta(t) + \theta \bar{\delta}(t-1) \\
 \alpha(t) &= \begin{cases} \alpha(t-1) + k & \text{if } \bar{\delta}(t) \delta(t) > 0 \\ (1-\Phi) \alpha(t-1) & \text{if } \bar{\delta}(t) \delta(t) < 0 \\ \alpha(t-1) & \text{if } \bar{\delta}(t) \delta(t) = 0 \end{cases}
 \end{aligned}$$

ATTACHMENT 1

Where:

$w(t)$ = a weight in the network

$\epsilon(t)$ = local learning rate for the weight

$\delta(t)$ = the element of the gradient associated with the weight

$\bar{\delta}(t)$ = weighted average of recent δ

$J(t)$ = total error in the network (e.g., sum of squared error over all inputs)

θ, Φ, k = constants controlling rate of learning

After each epoch (pass through all the training examples), the partial derivative of error with respect to each weight is calculated and multiplied by the local learning rate, and the weight is changed by that amount. If the current weight vector is in a trough parallel to one of the axes, this can be determined by the fact that the sign of the gradient in one direction keeps changing, while the sign of the gradient in another direction stays the same. The sign of the gradient will therefore often differ from the sign of the average of recent gradients. Once this is noticed, the local learning rate in the direction of the changing sign is decreased, and the rate in the direction of the constant sign is increased. This has the effect of slowing down wasteful movement across the trough, and speeds up movement along the trough. If the trough is aligned at a 45 degree angle to all the axes instead of parallel to one, then the signs of all the gradients will be constantly changing, and the weight vector takes small steps in the direction indicated by Backpropagation. This is unfortunate, but to compensate for this would require additional storage and computation time proportional to the square of the number of weights.

To see whether the sign of the gradient is changing, Delta-Bar-Delta keeps track of two things: the current gradient and an exponentially weighted sum of recent gradients. If these two have the same sign, then the local learning rate is increased, otherwise it is decreased. There was one final heuristic: when the local learning rate is raised, it is increased linearly by adding a constant on each time step. When it is lowered, it is decreased exponentially by dividing it on each time step by a constant. Thus the learning

ATTACHMENT 1

rate falls more quickly than it rises, and so when the nature of the error surface changes often, the weights will tend to change too slowly rather than too quickly, and previously learned information will be in less danger of being erased by momentarily large learning rates. The exponential decreasing also has the advantage of preventing a local learning rate from ever becoming zero or going negative, either of which would prevent correct operation of the algorithm.

5 EXPERIMENTS

In the experiments presented here, a number of different combinations of hybrid control system components are tested. Two variations of an indirect adaptive controller are used, both based on Time Delay Control [YI90]. Either the reduced canonical form of the plant is used, causing all the interesting dynamics to be compressed into a single scalar (described below in Section 5.1), or the full state vector form is used. The learning component can learn initially unmodeled dynamics as a function of both state, or as a function of state and control action. When it is a function of control action, then the partial derivative of the learned unmodeled dynamics with respect to control action is calculated, giving an improved estimate of the effect of control on state. Finally, the learning system can be constrained to learn only functions whose partial derivatives with respect to control action are constant (e.g., the control enters the governing dynamical equations linearly).

These various hybrid controllers are then compared relative to the problem of controlling a simulated plant having both spatial dependencies and noise. The controller should learn to control the plant in the presence of spatial dependencies wherever they occur. As the plant moves from one state to another, the unmodeled nonlinearities may appear in different ways. First, they might apply briefly in the middle of the transition from one region of state-space to another. If the effect is short-lived, then it will have a minimal impact on the trajectory of the plant. Also, once the plant leaves the region where the nonlinearity has an effect, it will have time to recover and move back towards the desired trajectory.

A more severe problem occurs if the nonlinearity appears and then remains present even after the state of the plant reaches the desired value. In this case, the nonlinearity has more time to affect the trajectory, and the plant may never leave its influence long enough to

ATTACHMENT 1

recover without adaptive or learning augmentation. If the nonlinearity is present throughout the plant's trajectory, then the problem is even more difficult. All three scenarios are considered in the experiments below.

Finally, the accuracy of the final controller is not the only issue to be considered. Since it is a learning system, it is also important to consider how fast it can learn, and how susceptible it is to forgetting one region while exploring another. These issues are examined by the experiments in the last section, below.

This chapter first describes the plant used for the simulations. The linear dynamical systems matrices are then derived for that plant, and the experimental results are presented for the hybrid system in various configurations. Finally, the Delta-Bar-Delta algorithm is compared with the standard Backpropagation algorithm, and then a modified Delta-Bar-Delta is examined.

All of the experiments below were based on a cart-pole plant being simulated at 50 Hz (using Euler integration), and a controller running at 10 Hz. The cart-pole system is shown figures 5.1 and 5.2. The *a priori* knowledge of the plant was limited to a linearized model of the system on the flat regions of the track. The 30 degree tilt in the region between 1 and 2 meters was completely unmodeled and had to be either adapted to or learned.

Unless otherwise noted, the learning system in all the experiments below was a Backpropagation, sigmoid, two layer network, with 10 nodes in each layer. Connections were made from the inputs to the first layer, from the first layer to the second, and from the second to the outputs. There were also connections from the first layer to the outputs. The inputs consisted of the four elements of state: cart position x , pole angle θ , cart velocity \dot{x} , and pole angular velocity $\dot{\theta}$. The network was trained using the unmodeled dynamics calculated by the adaptive TDC controller, while moving the cart to a new random position in the range 0 to 3 meters every 4 seconds. In the case of the reduced canonical form of the controller, the training was based on moving the cart from 0 to 3 meters and back, every 4

seconds.

5.1 THE CART-POLE SYSTEM

The plant used for the simulations is based on a standard inverted pendulum system. The problem is to move the cart to some desired track position by applying force directly to the cart center of mass, while at the same time balancing a pole that is attached to the cart via a hinge (see figures 5.1 and 5.2).

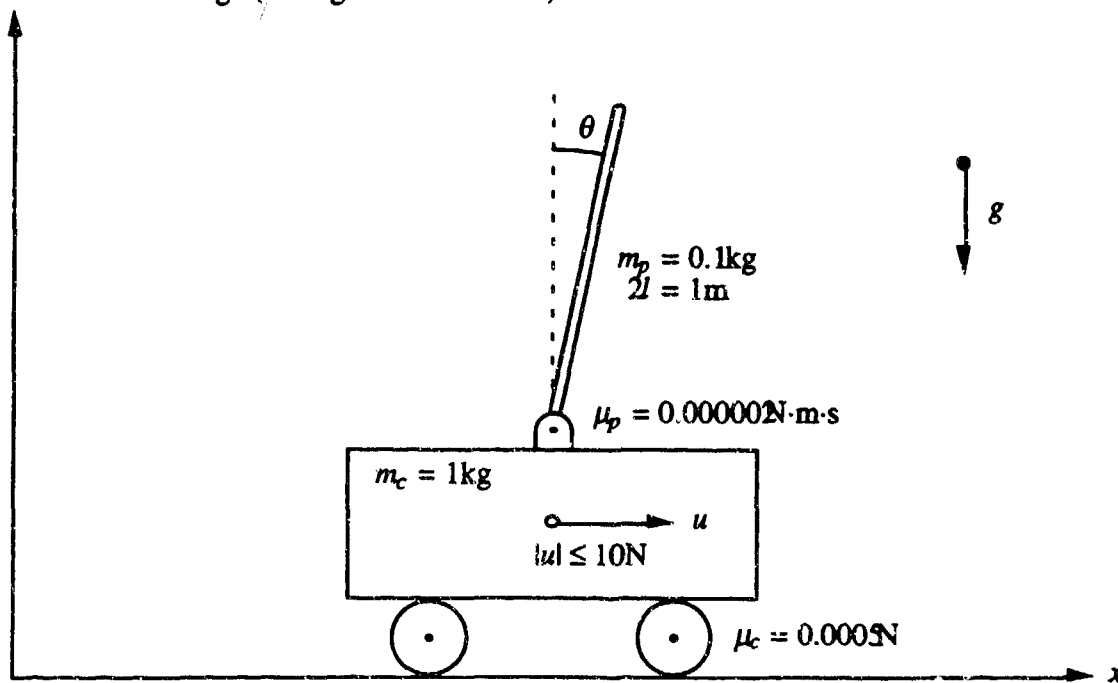


Figure 5.1 The cart-pole system

The design of an effective automatic control system for the cart-pole object on the split-level track is a challenging problem. The dynamical behavior of the nominal cart-pole system has the following attributes:

- nonlinear
- open-loop unstable
- nonminimum phase
- 4 state variables: $(x, \theta, \dot{x}, \dot{\theta})$

ATTACHMENT 1

The equations of motion for this plant are:

$$(m_c + m_p)\ddot{x} \sec \alpha + m_p l \ddot{\theta} \cos(\theta - \alpha) - m_p l \dot{\theta}^2 \sin(\theta - \alpha) - (m_c + m_p)g \sin \alpha = f - \mu_c \operatorname{sgn} \dot{x}$$

$$\frac{4}{3} m_p l^2 \ddot{\theta} + m_p l \ddot{x} \sec \alpha \cos(\theta - \alpha) - m_p g l \sin \theta = -\mu_p \dot{\theta}$$

where:

| | | |
|----------|---|---|
| x | = | position of the cart (m) |
| θ | = | pole angle (rad) |
| α | = | $\frac{\pi}{6}$ rad track incline angle |
| g | = | 9.8 m/s ² acceleration due to gravity |
| m_c | = | 1.0 kg mass of cart |
| m_p | = | 0.1 kg mass of pole |
| l | = | 0.5 m pole half-length |
| μ_c | = | 0.0005 N friction between cart and track |
| μ_p | = | 0.000002 N · m · s friction between pole and cart |
| $ f $ | ≤ | 10.0 N force applied to cart |

When the track angle is zero (horizontal track), both the equations of motion and the plant parameters are identical to those in [BB90] and [BSA83]. To test the learning ability of the system, one portion of the track is set on an incline, as shown in figure 5.2.

ATTACHMENT 1

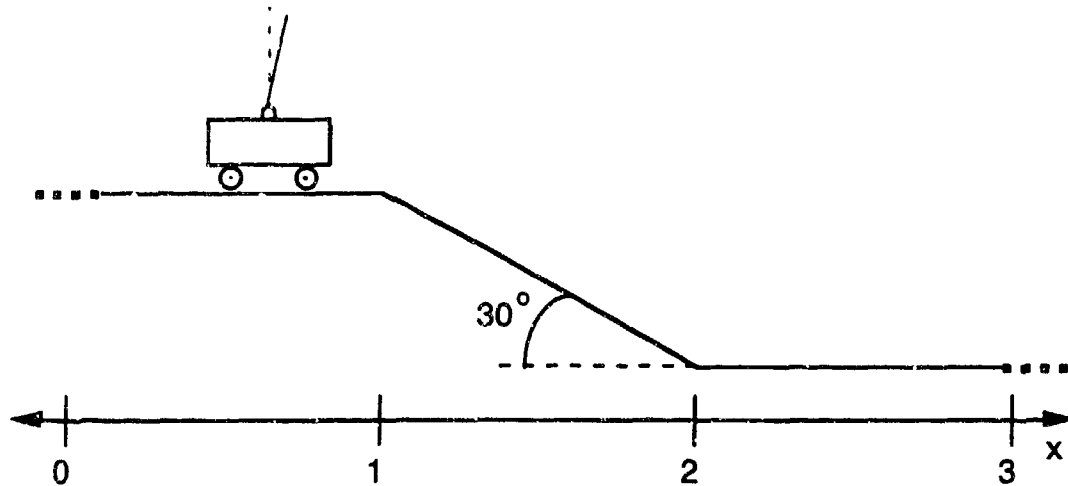


Figure 5.2 Cart-pole system

From the origin to 1 m, the track is level. From 1 m to 2 m, the track slopes down at a 30 degree angle towards the 2 m mark. From 2 m to 3 m the track is level again. The controller is given no *a priori* knowledge of the inclination of the track. It must adapt every time it reaches the incline, unless it eventually learns to anticipate it.

TDC allows *a priori* knowledge to be incorporated into the controller. Here, the *a priori* knowledge is a model formed by linearizing the actual plant equations about the origin, on the flat part of the track. Assuming small pole angles ($\theta \ll 1$) and a horizontal track ($\alpha = 0$), the equations-of-motion may be linearized, and the Laplace transform of them taken to yield a simple transfer function between force and cart position:

$$\frac{X(s)}{F(s)} = \frac{(s - 3.8360)(s + 3.8360)}{s^2(s - 3.9739)(s + 3.9739)}$$

The open-loop poles and zeros (the values of s where the above function is infinite or zero, respectively) are shown in Figure 5.3. The pole in the right-half plane causes it to be unstable: when left to itself, the pole on the cart generally falls. The zero in the right half-plane causes it to be nonminimum phase: thus to move the cart to the right when the pole is vertical, it is first necessary to move it a small amount to the left.

ATTACHMENT 1

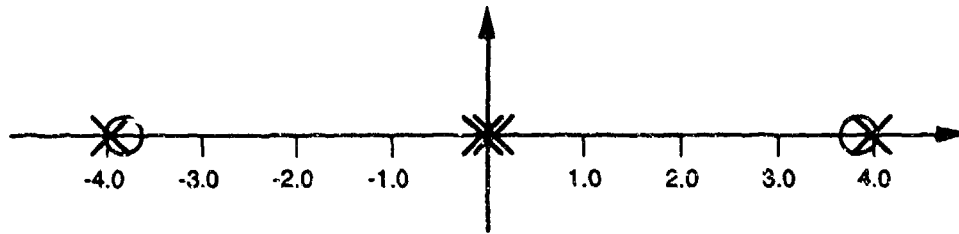


Figure 5.3 Open-loop poles and zeros in the complex plane

This linearized model is incorrect both in the tilted region of track and when the pole angle is large.

Taking the partial derivatives of the plant equations-of-motion and evaluating them at the origin yield a linear model of the plant. This model is of the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -0.7178 & 0 & 0 \\ 0 & 15.7917 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0.9756 \\ -1.4634 \end{bmatrix}$$

where the state vector $\mathbf{x} = (x, \theta, \dot{x}, \dot{\theta})$. The \mathbf{A} matrix indicates that the cart position and pole angle are the integrals of cart velocity and pole angular velocity respectively, and that the cart and pole velocities are both proportional to pole position. The \mathbf{B} matrix indicates that the force applied to the plant directly affects the cart and pole velocities. It is often more convenient to undertake a change of variables in the above equation to put it into *controller canonical form*. This form is found by first taking the original equations:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ y &= \mathbf{C}\mathbf{x} \end{aligned}$$

where y is the output being controlled. \mathbf{C} could be the identity vector, but for the plant being controlled here, \mathbf{C} is the new vector $[1 \ 0 \ 0 \ 0]$. A change of variables is then introduced by substituting $\mathbf{T}^{-1}\mathbf{x}$ for \mathbf{x} and rearranging the first equation to get:

ATTACHMENT 1

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{TAT}^{-1}\mathbf{x} + \mathbf{TB}u \\ y &= \mathbf{T}^{-1}\mathbf{C}\mathbf{x}\end{aligned}$$

These new equations are then treated as a new plant, with the "A" matrix of the new plant being \mathbf{TAT}^{-1} and the "B" matrix of the new plant \mathbf{TB} . The new plant is input/output equivalent to the original, one since varying u will have the same effect on y as in the original plant. The purpose of this change of variables is to convert the "A" and "B" matrices to the more convenient form:

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}_c\mathbf{x} + \mathbf{B}_cu \\ y &= \mathbf{C}_c\mathbf{x}\end{aligned}$$

$$\mathbf{A}_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 15.7917 & 0 \end{bmatrix}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{C}_c = [-14.3561 \quad 0 \quad 0.9756 \quad 0]$$

This controller canonical form has three important properties: the \mathbf{B}_c matrix is all zeros with a 1 at the bottom, the \mathbf{A}_c matrix without its first column and last row is the identity matrix, and the first column of the \mathbf{A}_c matrix is all zeros except possibly for the bottom position. The bottom row of the \mathbf{A}_c matrix could have been anything, and it would still have been in canonical form. For the \mathbf{x} vector in this new canonical form, the first element is the integral of the second, the second element is the integral of the third, the third element is the integral of the fourth, and the last element is a linear function of all elements. The control action u only directly affects the fourth element of the state. This form is convenient because a reduced version of the TDC control law can be developed from it that involves scalar and vector algebra and vector inner products instead of full vector-matrix algebra. In particular the matrix inversion (or pseudo-inversion) required by the original control law reduces to a simple scalar division operation. In addition, the learning system will only have to learn a scalar output instead of a four element vector.

ATTACHMENT 1

The only complicated part of the above transformation was choosing T^{-1} . This can be done in MATLAB™ with the following code:

$$d = \text{poly}(A)$$

$$T_{\text{inv}} = \text{ctrb}(A, B) * \text{hankel}(d(\text{length}(d) - 1 : -1 : 1))$$

where, if A is n by n , then d is a row vector with $n+1$ elements. The function $\text{poly}(A)$ returns the coefficients of the characteristic equation of A , which is the polynomial formed by the determinant of $(\lambda I - A)$. The expression " $d(\text{length}(d) - 1 : -1 : 1)$ " removes the first element of d , the lowest order coefficient, and reverses the remaining elements. The function hankel returns an n by n matrix that has its first column equal to this list and all zeros below the first anti-diagonal. Each element of the matrix equals the element one below and to the left of it. Finally, ctrb returns the n by n controllability test matrix (a row of column vectors) formed from the n by n matrix A and the n by 1 vector B by:

$$\text{ctrb}(A, B) = [B \quad AB \quad A^2B \quad A^3B \quad \dots \quad A^{n-1}B]$$

In discrete-time, the full system is approximated by:

$$x_{k+1} = \Phi x_k + \Gamma u_k$$

At 50 Hz:

$$\Phi_{50} = \begin{bmatrix} 1.000 & -0.0001 & 0.02 & 0 \\ 0 & 1.0032 & 0 & 0.0200 \\ 0 & -0.0144 & 1 & -0.0001 \\ 0 & 0.3162 & 0 & 1.0032 \end{bmatrix} \quad \Gamma_{50} = \begin{bmatrix} 0.0002 \\ -0.0003 \\ 0.0195 \\ -0.0293 \end{bmatrix}$$

At 10 Hz:

$$\Phi_{10} = \begin{bmatrix} 1.000 & -0.0036 & 0.1000 & -0.0001 \\ 0 & 1.0800 & 0 & 0.1027 \\ 0 & -0.0737 & 1.0000 & -0.0036 \\ 0 & 1.6211 & 0 & 1.0800 \end{bmatrix} \quad \Gamma_{10} = \begin{bmatrix} 0.0049 \\ -0.0074 \\ 0.0977 \\ -0.1502 \end{bmatrix}$$

The behavior of the reference model, in discrete time, is given by:

ATTACHMENT 1

$$x_{k+1} = \Phi_m x_k + \Gamma_m u_k$$

At 50 Hz:

$$\Phi_{M50} = \begin{bmatrix} 1.0002 & 0.0048 & 0.0204 & 0.0012 \\ -0.0003 & 0.9958 & -0.0005 & 0.0182 \\ 0.0184 & 0.4712 & 1.0343 & 0.1201 \\ -0.0276 & -0.4129 & -0.0515 & 0.8226 \end{bmatrix} \quad \Gamma_{M50} = \begin{bmatrix} -0.0002 \\ 0.0003 \\ -0.0184 \\ 0.0276 \end{bmatrix}$$

At 10 Hz:

$$\Phi_{M10} = \begin{bmatrix} 1.0038 & 0.1077 & 0.1072 & 0.0276 \\ -0.0058 & 0.9108 & -0.0109 & 0.0606 \\ 0.0654 & 2.0162 & 1.1249 & 0.5176 \\ -0.1012 & -1.5989 & -0.1931 & 0.2770 \end{bmatrix} \quad \Gamma_{M10} = \begin{bmatrix} -0.0038 \\ 0.0058 \\ -0.0654 \\ 0.1012 \end{bmatrix}$$

The error feedback gain K is zero. This means that, given the plant's state at time k , the desired state for the plant at time $k+1$ will always be equal to the state that the reference model would have at time $k+1$ if it started at the state where the plant is at time k . In other words, for a given commanded state, there will be a set of almost parallel trajectories through state-space, which are the paths that the reference model would take. At any given point in time, the desired dynamics for the plant is simply to follow the reference trajectory that it is currently on. If K was greater than zero, then the desired dynamics of the plant could be faster than the dynamics of the reference model. The controller would then have to maintain a reference model internally. On the first time step, the state of the reference would be set to the state of the plant. On each time step thereafter, the reference model would be updated according to the reference dynamics. If the plant state matched the reference state, the desired next state of the plant would be equal to the desired next state of the reference. If the plant ever got off of the reference path, then it would not start following a new reference path, but would instead try to get back on to the original path. This integrating kind of behavior acts to keep small errors in the controller from building up over time. Although the added complexity of a nonzero K is never used in the experiments presented here, it would be easy to add the terms for a nonzero K into the hybrid controller. In fact, the equations derived above explicitly contain the terms for K , even though they are never used here.

5.2 ORGANIZATION OF THE EXPERIMENTS

The cart-pole track is horizontal everywhere except between the points at 1 meter and 2 meters. The 30 degree incline in this region is a large, unmodeled nonlinearity, and so is a more difficult region for the controller unless the learning component is working well. When the cart starts at 0 meters and is told to move to 3 meters, most of the complicated maneuvering and acceleration will be executed near the start and end of the trajectory, both of which are on the well-modeled level part of the track. This trajectory is therefore easier for the adaptive controller than moving from 0.8 to 1.3 meters, where it would have to cross the border of the nonlinearity almost immediately, and would then have to stop on the incline near the edge. The following sections are organized around trajectories of differing difficulty: (i) nonlinearities in the middle, (ii) at the end, or (iii) at the start, middle, and end.

In all experiments, the inclined portion of the track is between the 1 and 2 meter mark. Section 5.4 shows results for the cart moving from 0 meters to 3 meters. Section 5.5 shows results for the trajectory from 0 to 1.3. Section 5.6 shows results when going from 0.8 to 1.3, and also for going from 1.3 to 1.9.

The networks were trained from data generated as the cart was commanded every 4 seconds to move to a new random position between 0 meters and 3 meters. The graphs show the performance of the hybrid over a 9 second period, after learning had already occurred. Two hybrid systems are compared: the reduced hybrid, which learns a scalar version of the unmodeled dynamics associated with the canonical system model described above, and the full hybrid, which learns the vector form of the unmodeled dynamics.

In sections 5.3, 5.4, and 5.5, the full hybrid uses input/output partial derivative information from a network that is constrained to have an output calculated as a general nonlinear function of x , and a constant, linear function of u . This network was used because it was found to give better performance than a network calculating outputs as a

general, nonlinear function of both x and u . For the sake of comparison, one run of the general network is shown in section 5.6. There is also one experiment shown in section 5.6 for the case of extremely noisy sensors, which is included to demonstrate that both of the hybrid controllers can continue to work under extremely noisy conditions.

The results are shown throughout this chapter in a consistent format. The position graphs show the position of the reference cart on the track in meters, as well as the position of the cart controlled by the full and reduced hybrid controllers. The other type of graph shows the error in position (reference minus actual) in meters, and the force applied. The force is scaled by a factor of ten, so that the range of the graph corresponds to the full ± 10 N range of admissible forces that can be applied to the cart-pole system.

5.3 MID-TRAJECTORY SPATIAL NONLINEARITIES

The first set of experiments were intended to test the ability of the hybrid controllers in the presence of spatial dependencies appearing in the middle of the plant's trajectory. In addition to inherent nonlinearities in the cart-pole system, a further nonlinearity was added by tilting the track 30 degrees in the region from 1 meter to 2 meters. In these first experiments, the cart was commanded to move from its initial position at 0 meters to a final position at 3 meters, while following a desired trajectory through state-space, and without allowing the pole to fall over. Since it spent relatively little time in the inclined region, and since it always left that region before it came close to the final state, this setup introduced mid-trajectory spatial nonlinearities.

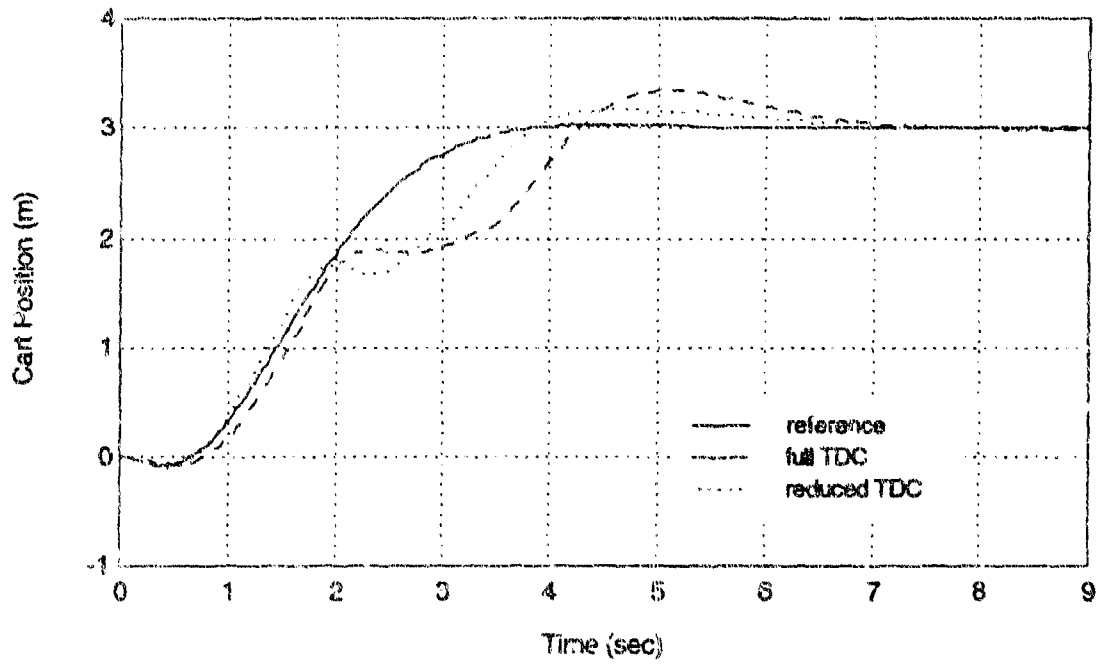


Figure 5.4 Plain TDC, from 0 to 3 meters

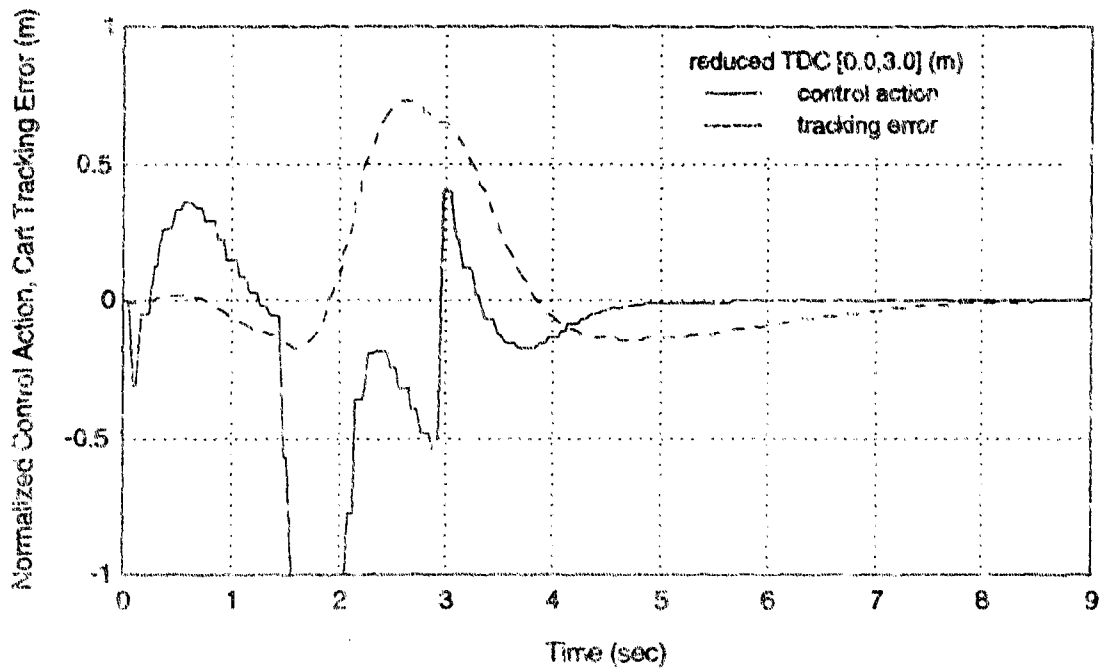


Figure 5.5 Reduced TDC; force and position error

ATTACHMENT 1

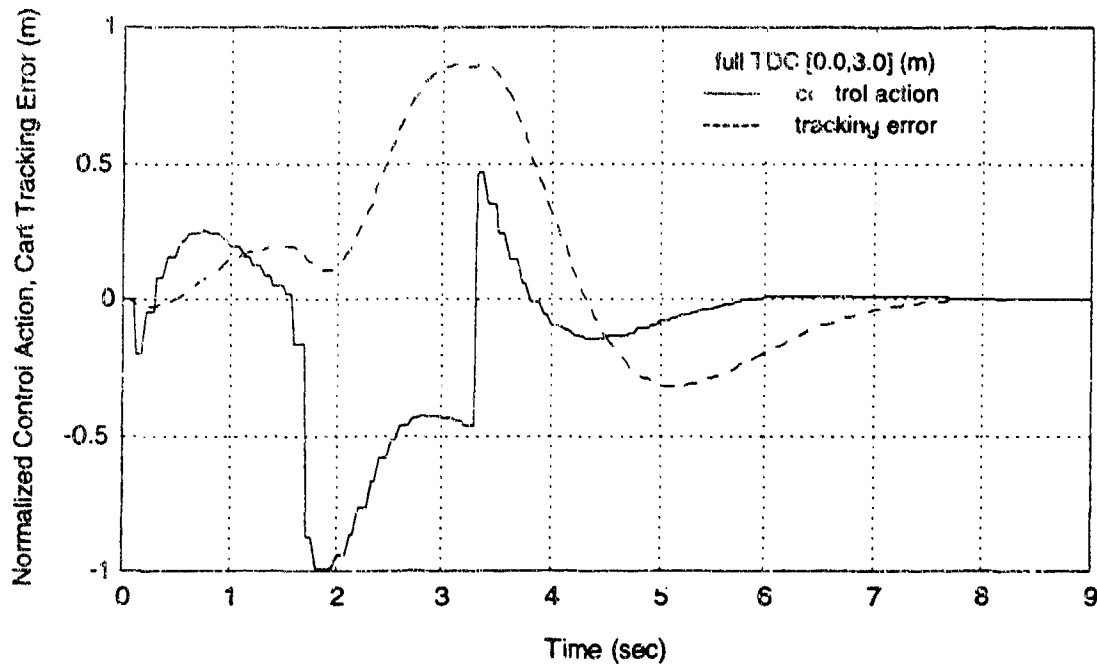


Figure 5.6 Full TDC; force and position error

Figure 5.4 demonstrates the difficulty of this control task for TDC alone, without learning. Both the reduced and full versions of TDC are able to balance the pole, but they do not follow the desired trajectory very closely. For the reduced version, figure 5.5 shows that there were not very large errors in the cart position until after the actuator started to saturate at -10 N. If it could have applied more than that level of force, it might have done better. The full TDC had equally bad errors, but did not attempt to apply more force than was possible.

Figures 5.7, 5.8, and 5.9 depict the same experiment, but with the hybrid controller.

ATTACHMENT 1

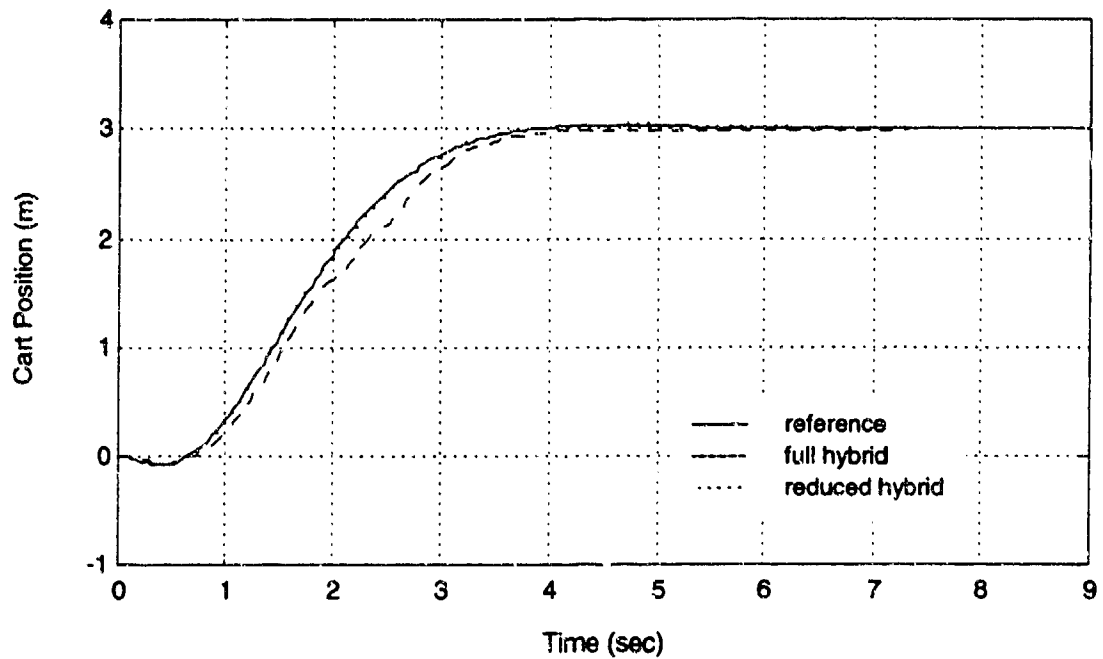


Figure 5.7 Full and reduced hybrid systems, from 0 to 3 meters

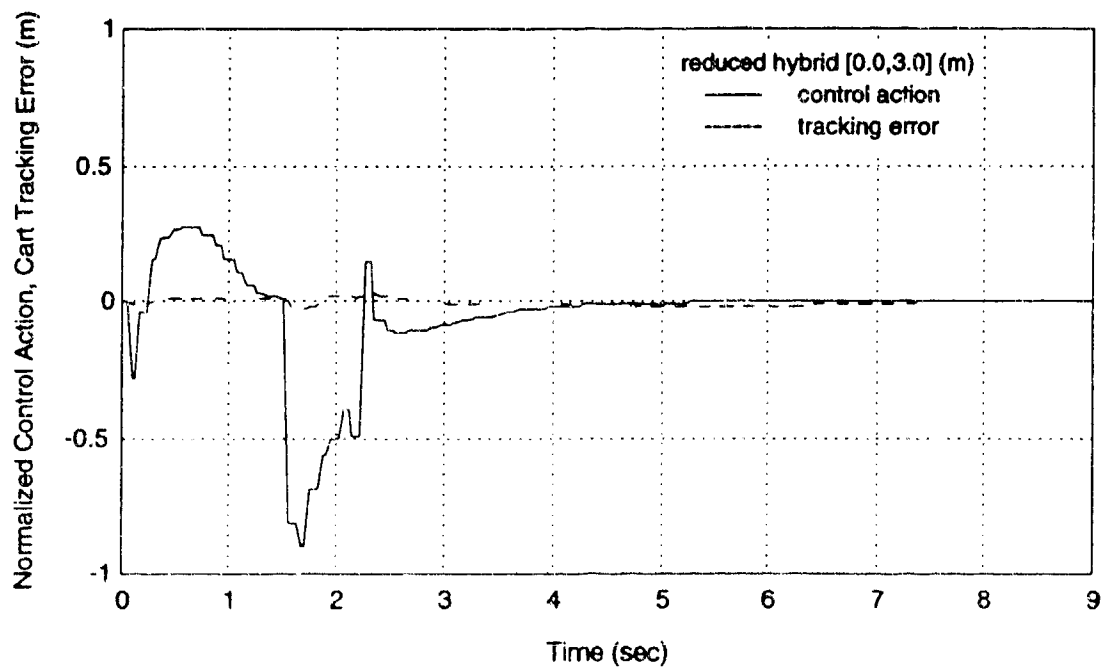


Figure 5.8 Reduced hybrid, from 0 to 3 meters; force and position error

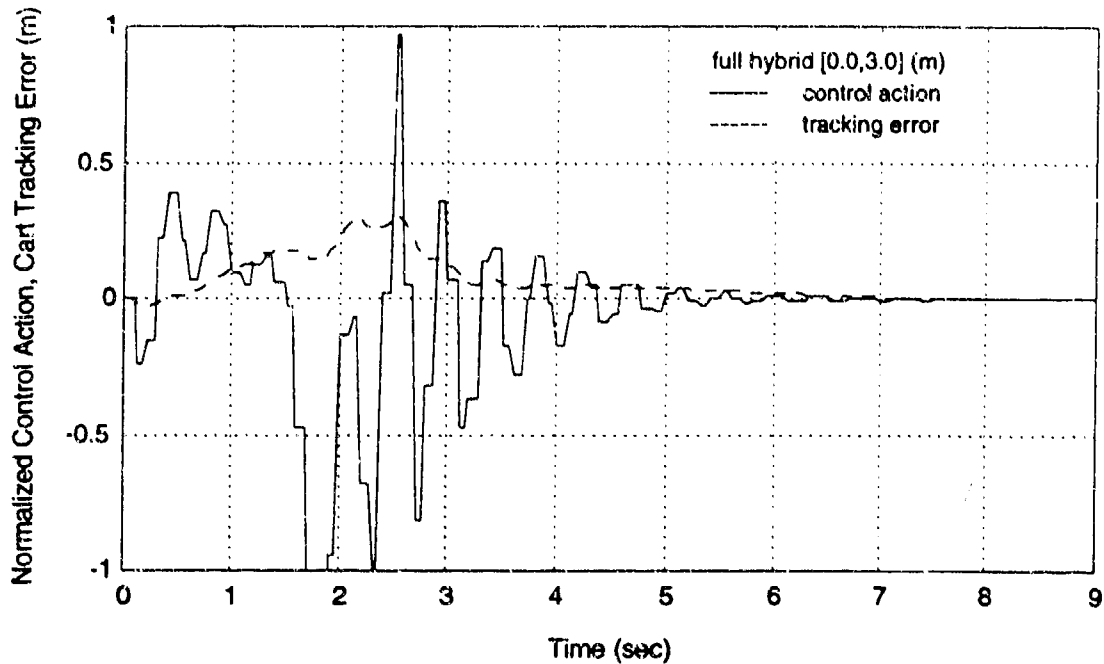


Figure 5.9 Full hybrid, from 0 to 3 meters; force and position error

With the aid of learning, the hybrid controllers performed extremely well. The reference and actual trajectories were almost completely on top of each other, and appear to be a single curve. The full hybrid is comparable to the reduced version, although the reduced version tracked the reference slightly better. It is interesting that although the full TDC attempted to use less force than the reduced TDC, the full hybrid applied more control action than the reduced hybrid. In fact, the full hybrid tends to oscillate in its application of control action, even though the cart itself did not oscillate visible.

The experiments in this section demonstrated three things. First, an adaptive controller can be improved significantly when used in a hybrid architecture with a learning system. Second, in some cases, such as the reduced canonical form shown here, simply learning unmodeled dynamics is enough to give acceptable performance. In other cases, such as in the full (noncanonical) form, the performance is not very good unless the input/output partial derivatives of the learned function are also used, and the network itself is modified for this purpose.

5.4 TRAJECTORY-END SPATIAL NONLINEARITIES

The simulations in sections 5.3 were all conducted while commanding the cart to move from 0 meters to 3 meters. Since the unexpected tilt in the track was between 1 and 2 meters, the learning system was mainly beneficial during the brief period that the track was on the incline. Any errors introduced into the state during that period can be handled after the plant has moved on to a region where its *a priori* model is more accurate. A more difficult problem occurs when the cart is commanded to move from 0 meters to 1.3 meters. Then the spatial dependencies are important at the end of the trajectory, when the cart should be decelerating and settling in on the final state. This section compares the behavior of the reduced and full TDC and hybrid controllers in this more difficult situation.

Figures 5.10, 5.11, and 5.12 show plain TDC trying to move the cart from 0 to 1.3 meters. Both controllers are fine until they reach the edge of the incline at 1 meter. At this point they are trying to decelerate since they are near the goal. The unexpected acceleration causes the pole to fall back, and the cart must then back up past the edge to keep it from falling. This sets up the oscillations around the 1 meter mark which are visible in the figures. The reduced canonical form TDC eventually allows the pole to fall over, while the full TDC eventually reaches the goal, but only after 10 seconds of oscillations. This is exactly the kind of situation for which the integration with the learning system would be expected to be most valuable.

Figures 5.13, 5.14, and 5.15 show the hybrid controllers performing much better on the same problem. Not only is the performance better, but it is accomplished using less force, and saturating less often.

ATTACHMENT 1

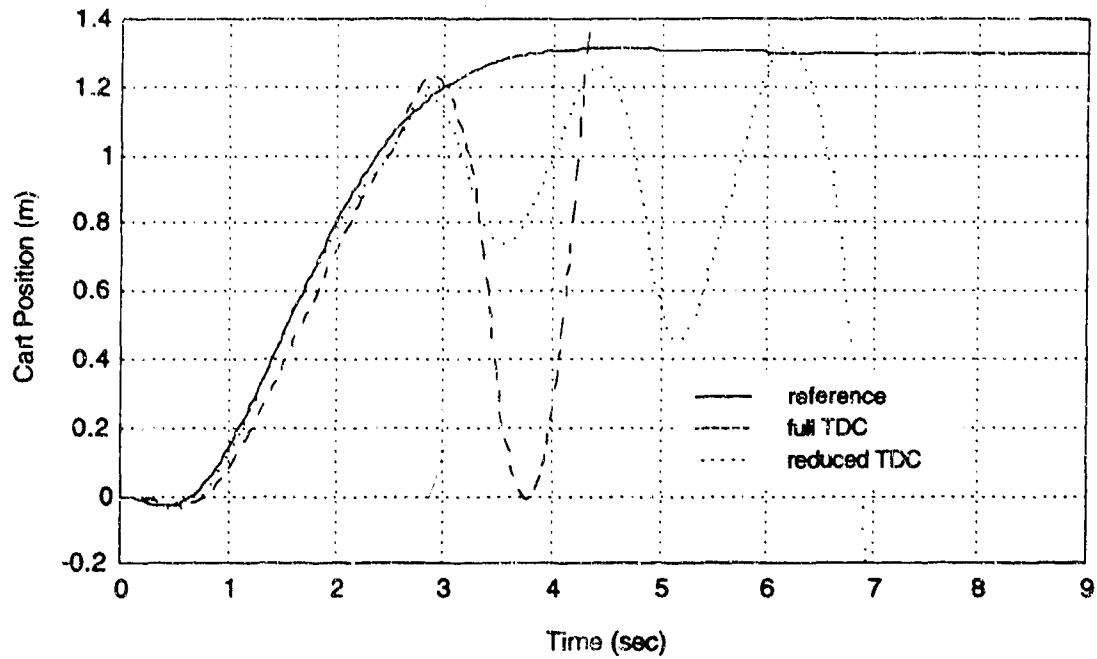


Figure 5.10 Plain TDC, from 0 to 1.3 meters

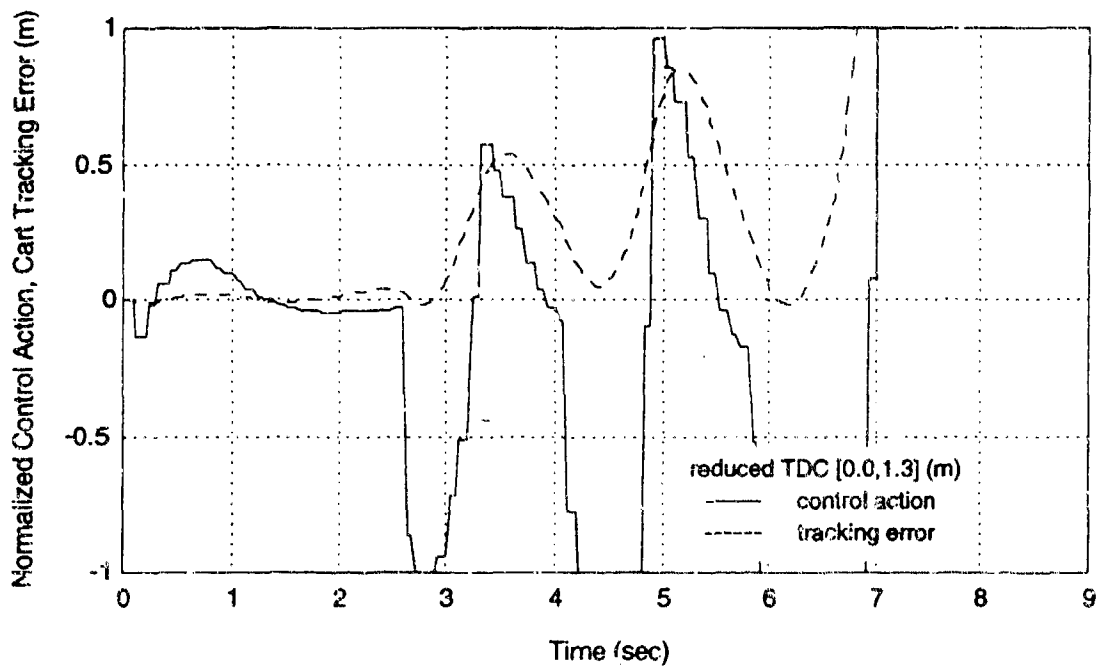


Figure 5.11 Plain TDC, from 0 to 1.3 meters; force and position error

ATTACHMENT 1

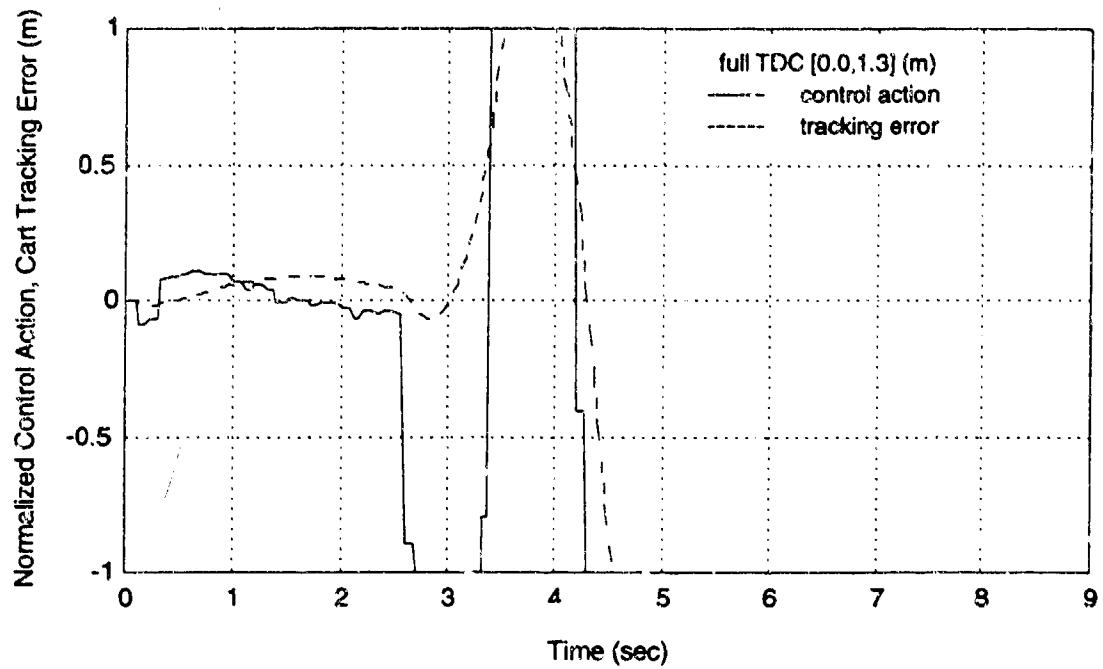


Figure 5.12 Plain TDC, from 0 to 1.3 meters; force and position error

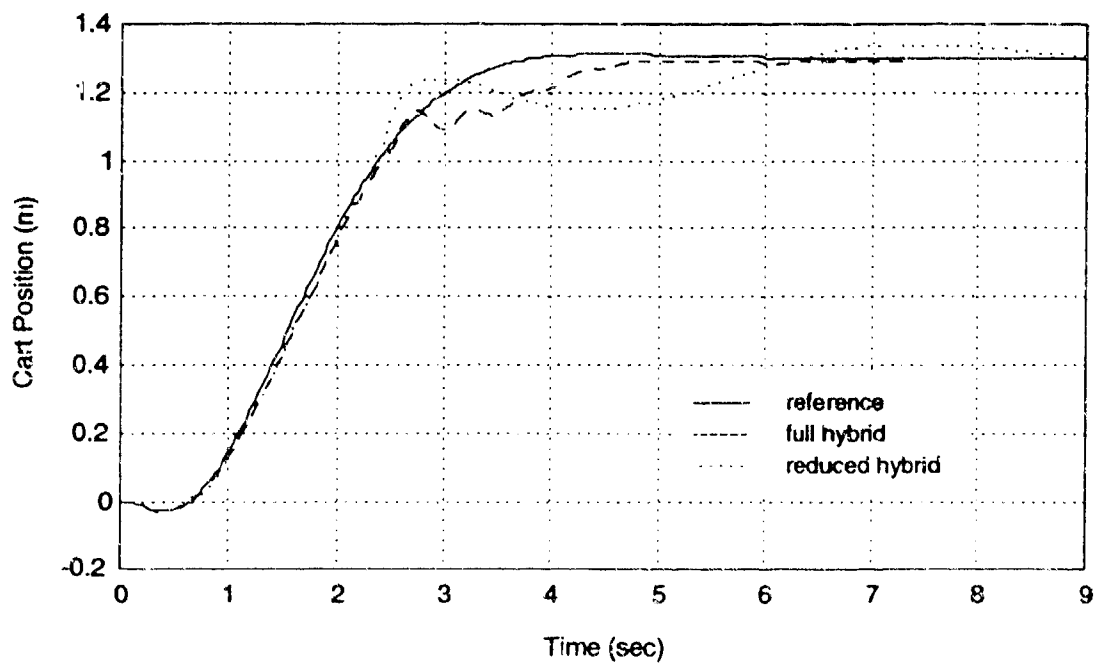


Figure 5.13 Full and reduced hybrid systems; from 0 to 1.3 meters

ATTACHMENT 1

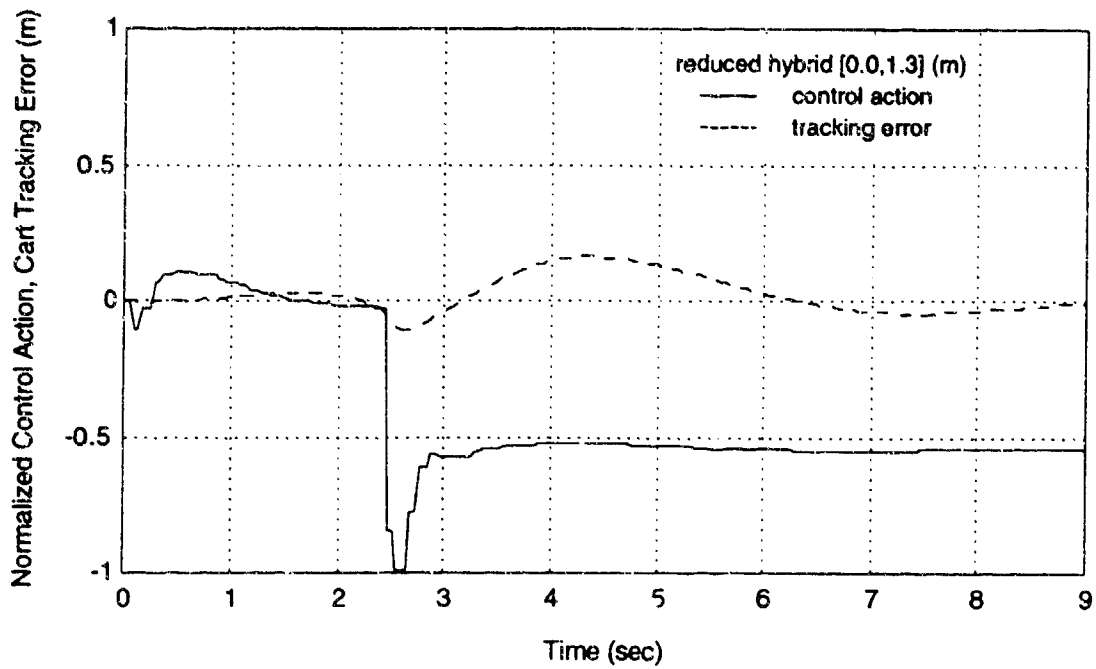


Figure 5.14 Reduced hybrid, from 0 to 1.3 meters; force and position error

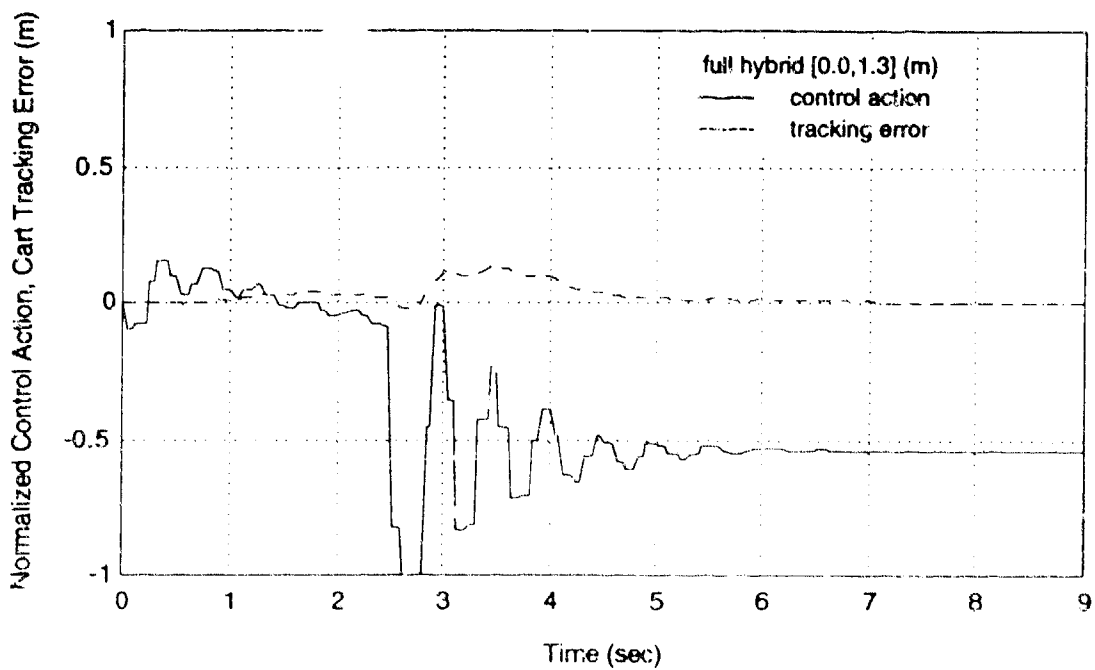


Figure 5.15 Full hybrid, from 0 to 1.3 meters; force and position error

In this problem, the full hybrid follows the reference more closely and overshoots less than the reduced hybrid. It is not surprising that the full controller is better than the

ATTACHMENT 1

reduced one in this case, but not in the case of moving from 0 to 3 meters. When the nonlinearity affected the trajectory only for a brief period of time in the middle, any learning system that could predict that nonlinearity at all could do a good job. However, in the more demanding problem of stopping the cart on the incline near the edge, the exact nature of the nonlinearities on the slope become more important. In this case, it is more important to get better estimates of the effect of control on state, by using the partial derivatives of the function that was learned.

5.5 TRAJECTORY-START AND TRAJECTORY-END NONLINEARITIES

It has been shown above that there is a performance improvement created by using partial derivative information in the hybrid. A more difficult control problem arose when the transition in or out of the tilted region occurred near the end of the trajectory, since that is the point that the cart starting to slow down and settle in to the correct position. The improvement from using partial derivative information was even more pronounced in this more difficult problem. In this section, a yet more difficult problem is considered, where the cart is commanded to move from 1.8 meters to 2.3 meters. This trajectory is short, so when the cart crosses the boundary of the tilted region, this event is both near the start of the run and near the end of it. Figures 5.16, 5.17 and 5.18 compare the behavior of the reduced and full hybrid controllers. The commanded path was from 0.8 meters to 1.3 meters.

ATTACHMENT 1

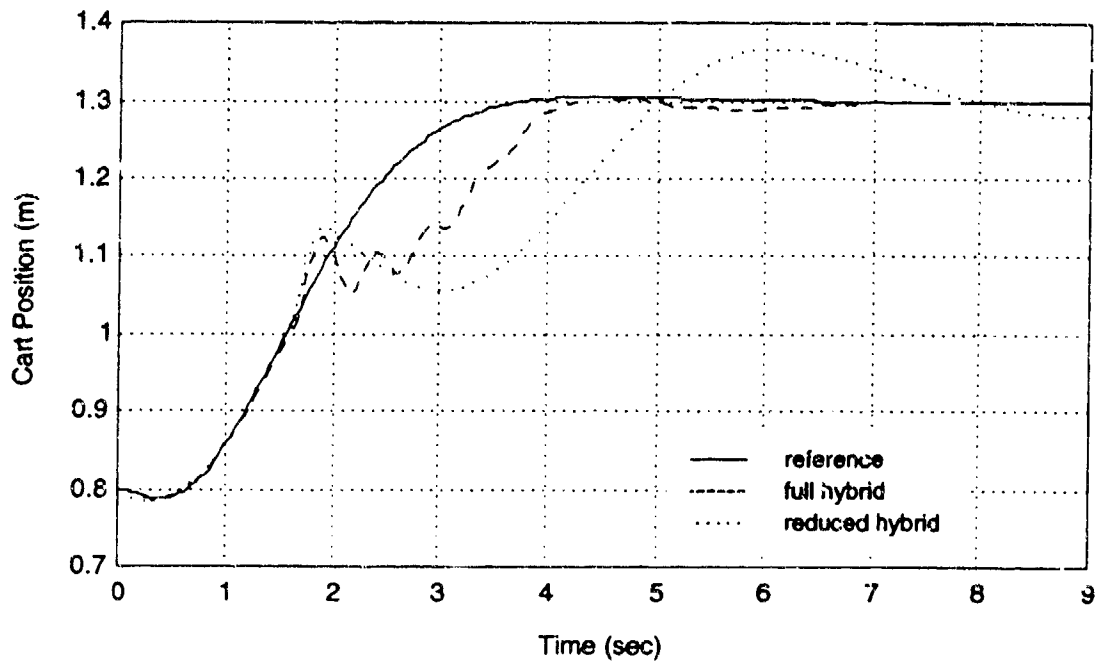


Figure 5.16 Full and reduced hybrid systems; from 0.8 to 1.3 meters

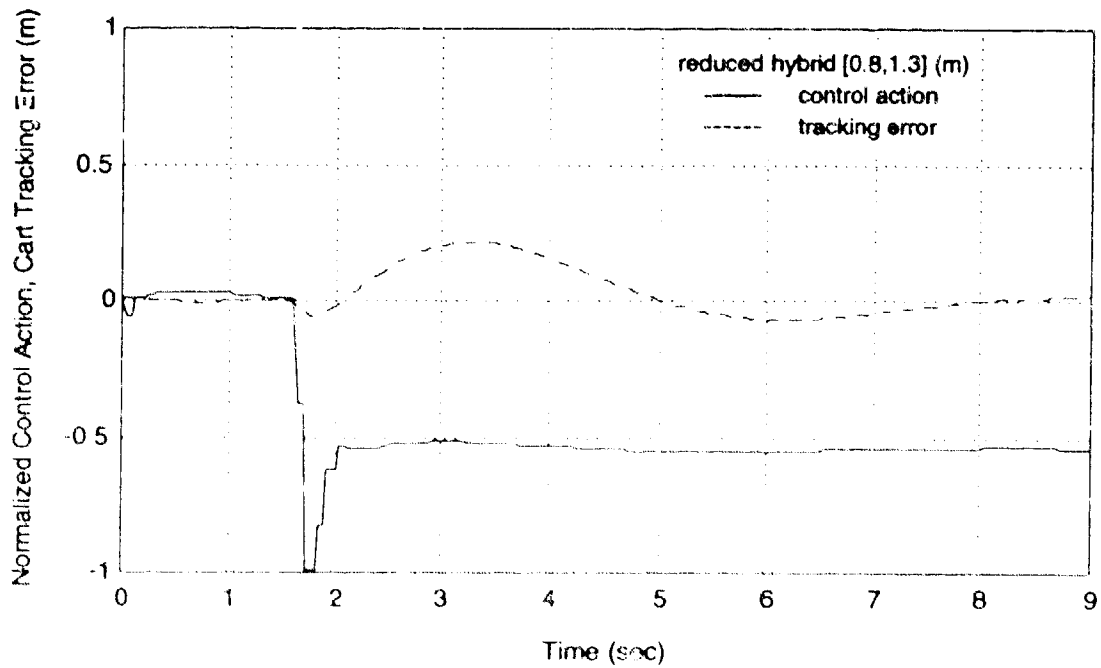


Figure 5.17 Reduced hybrid, from 0.8 to 1.3 meters; force and position error

ATTACHMENT 1

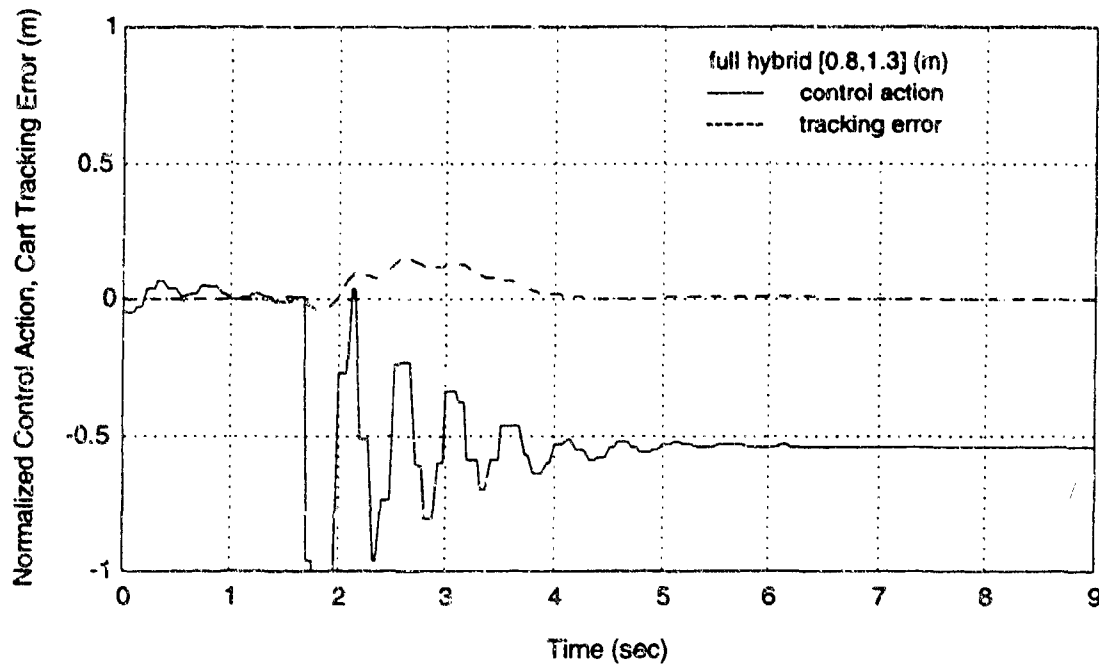


Figure 5.18 Full hybrid, from 0.8 to 1.3 meters; force and position error

As expected, the incorporation of the partial derivative information has a more dramatic effect here than it did in the previous problems. Figure 5.16 shows no overshoot at all for the full hybrid, as compared to a large overshoot for the reduced hybrid. As before, the force applied by the full hybrid was greater than the force applied by the reduced controller.

ATTACHMENT 1

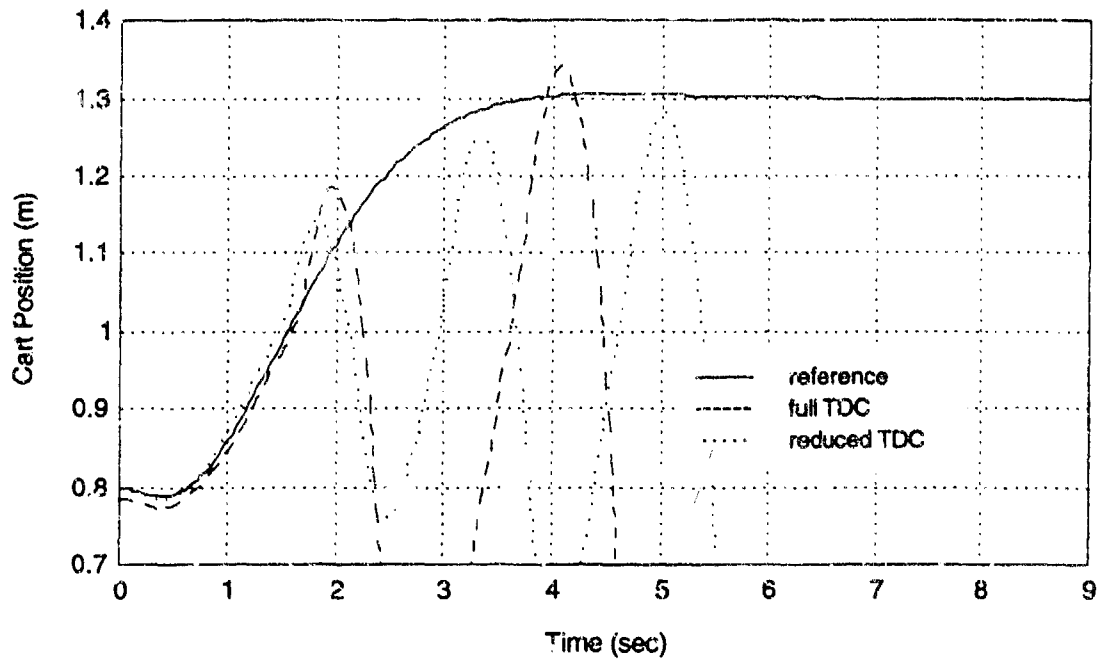


Figure 5.19 Plain TDC, from 0.8 to 1.3 meters

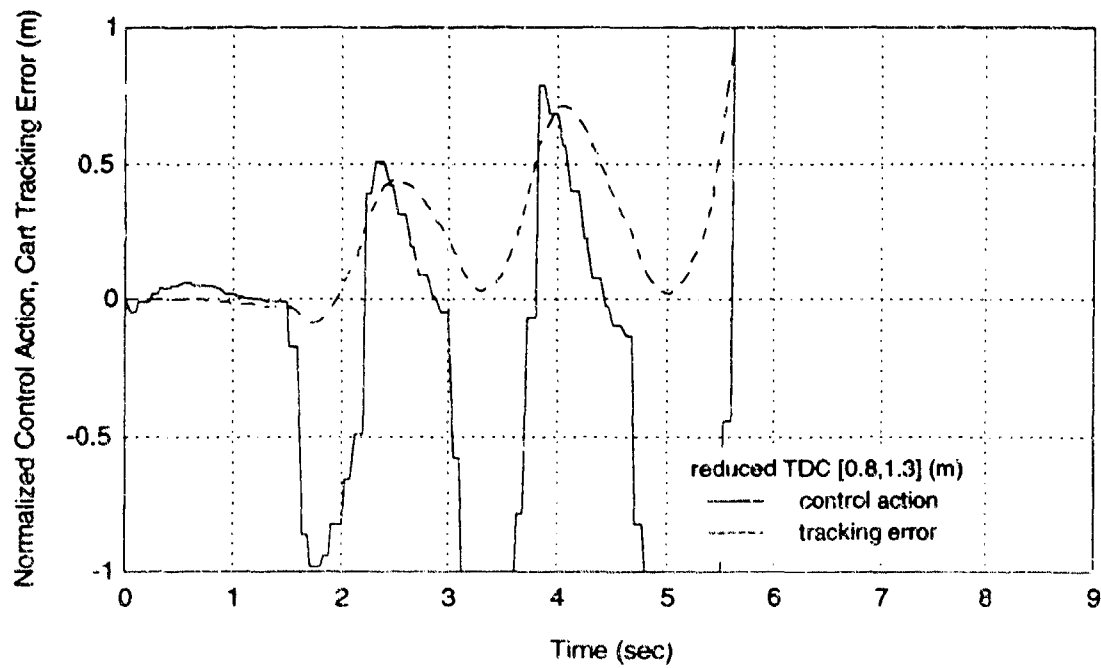


Figure 5.20 Reduced TDC, from 0.8 to 1.3 meters; force and position error

ATTACHMENT 1

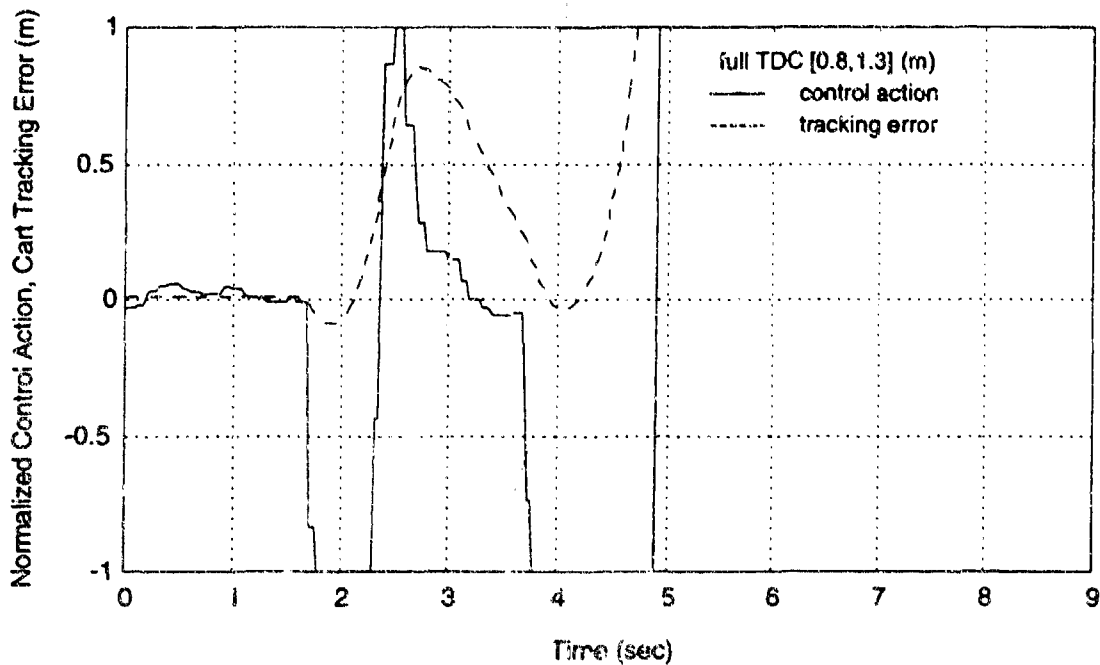


Figure 5.21 Full TDC, from 0.8 to 1.3 meters; force and position error

The performance of the hybrid is more impressive when compared with the result of plain TDC, as shown in figures 5.19, 5.20, and 5.21. Not only were the oscillations extreme, but the pole actually fell over after 5 or 6 seconds.

The same experiment was repeated commanding the controller to go from 1.3 meters to 1.9 meters. This ensured that the entire trajectory was on the inclined region of the track, and so the learning component was very important. The performance of the hybrid is shown in figures 5.22, 5.23, and 5.24.

ATTACHMENT 1

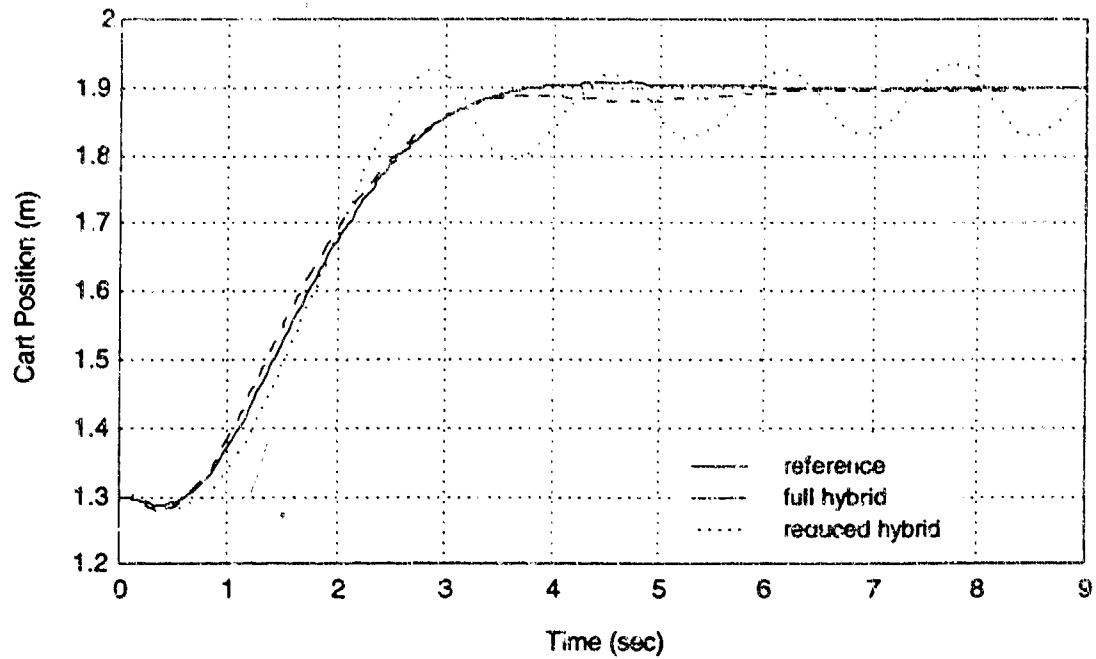


Figure 5.22 Full and reduced hybrid; from 1.3 to 1.9 meters

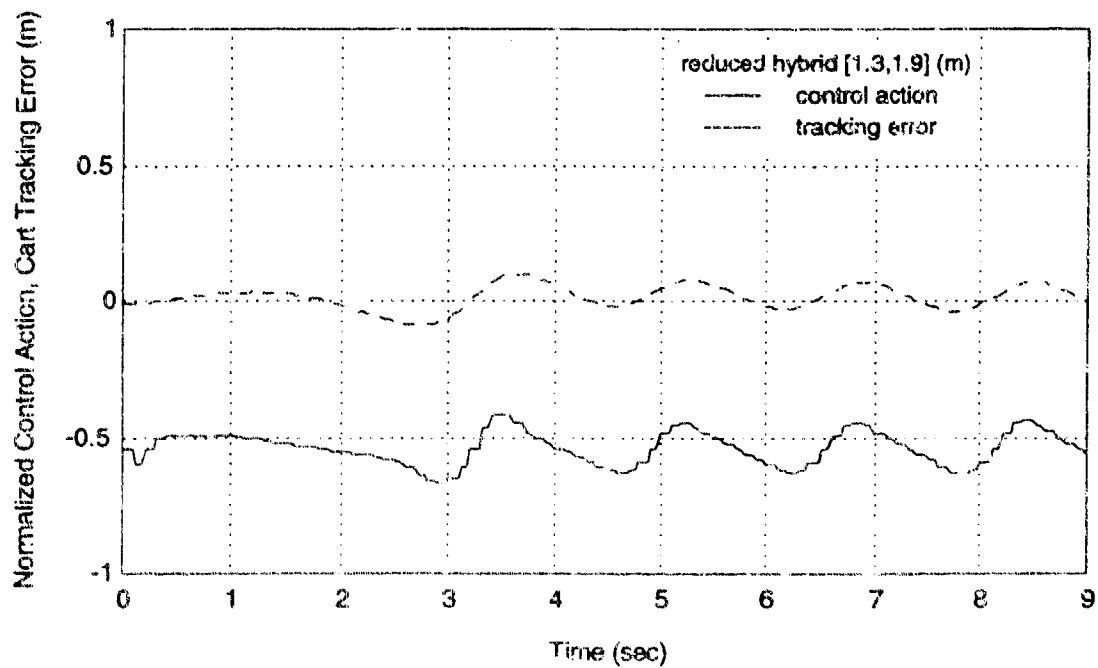


Figure 5.23 Reduced hybrid, from 1.3 to 1.9 meters; force and position error

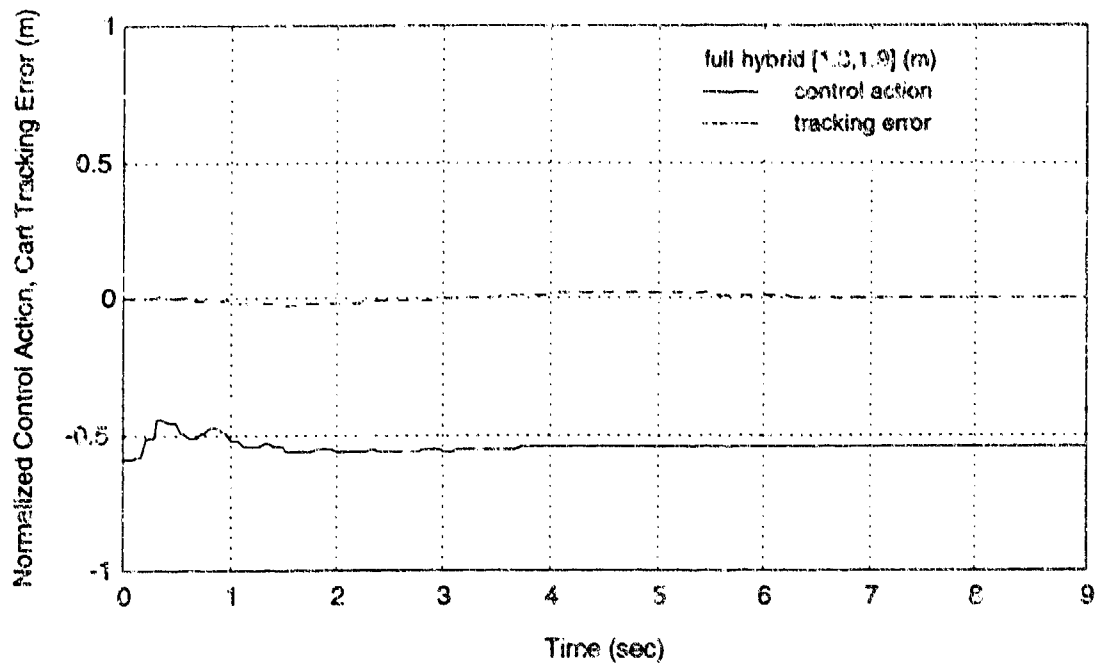


Figure 5.24 Full hybrid, from 1.3 to 1.9 meters; force and position error

The value of the extra partial derivative information in the full hybrid controller is exceptionally clear in figure 5.22. The full hybrid gives very acceptable performance, while the reduced hybrid actually goes into a limit cycle that continues indefinitely. This is due to the fact that small errors made near the edge of the incline tend to cause the cart to go across the boundary, thus greatly increasing the errors and inducing further crossings and further errors. The final results, in figure 5.25, 5.26, and 5.27, are the graphs for the same experiment with just plain TDC and no learning.

ATTACHMENT 1

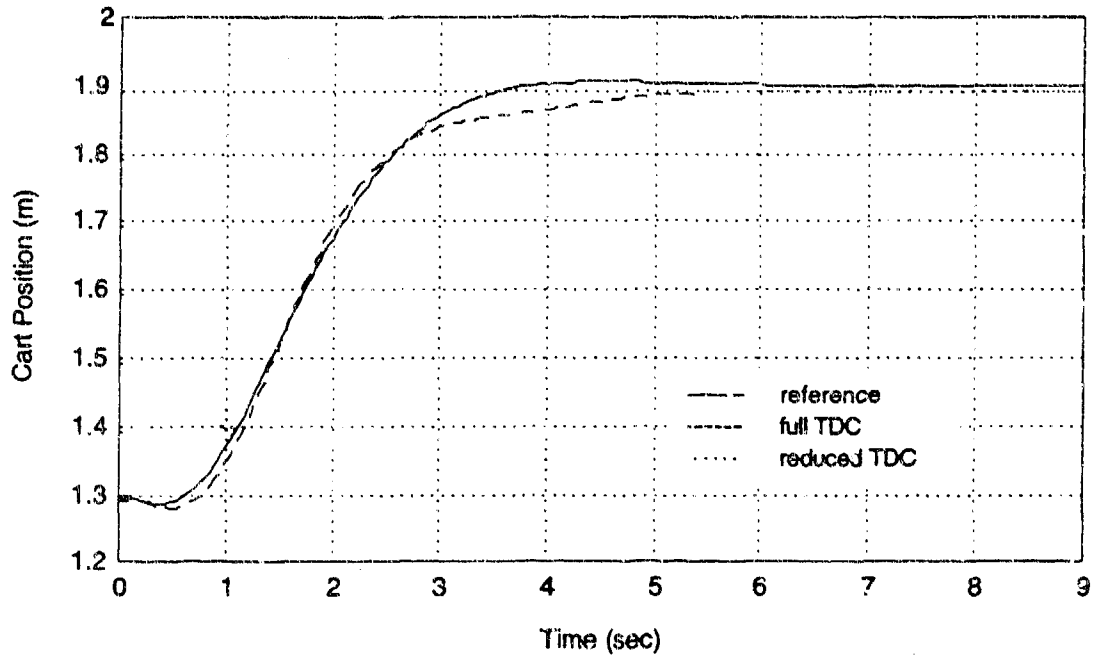


Figure 5.25 Plain TDC, from 1.3 to 1.9 meters

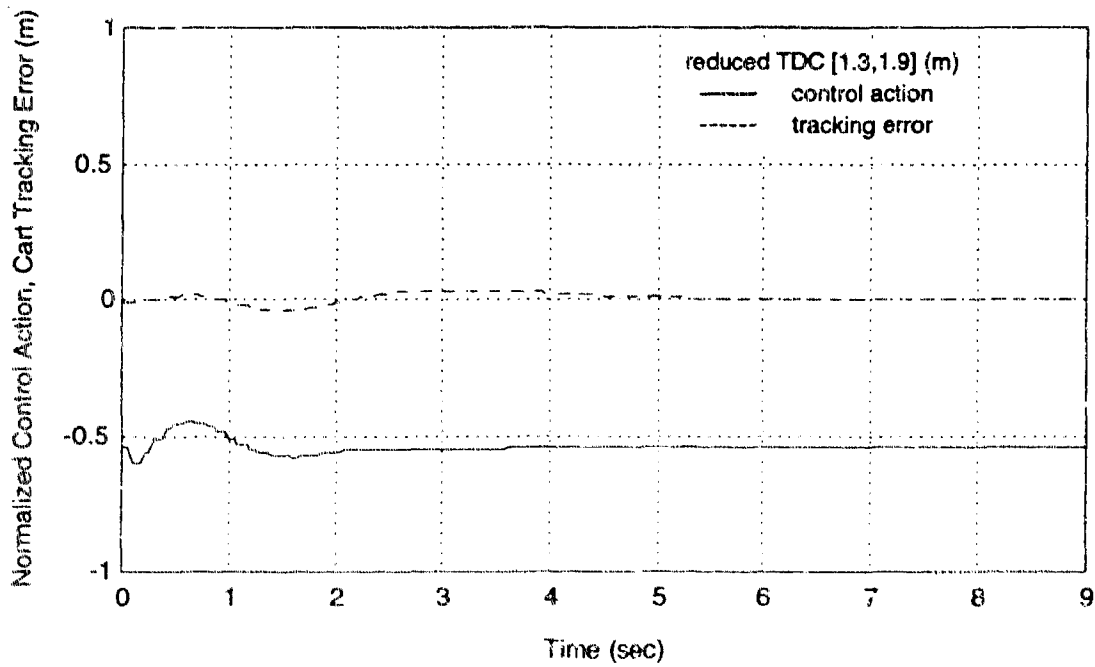


Figure 5.26 Reduced TDC, from 1.3 to 1.9 meters; force and position error

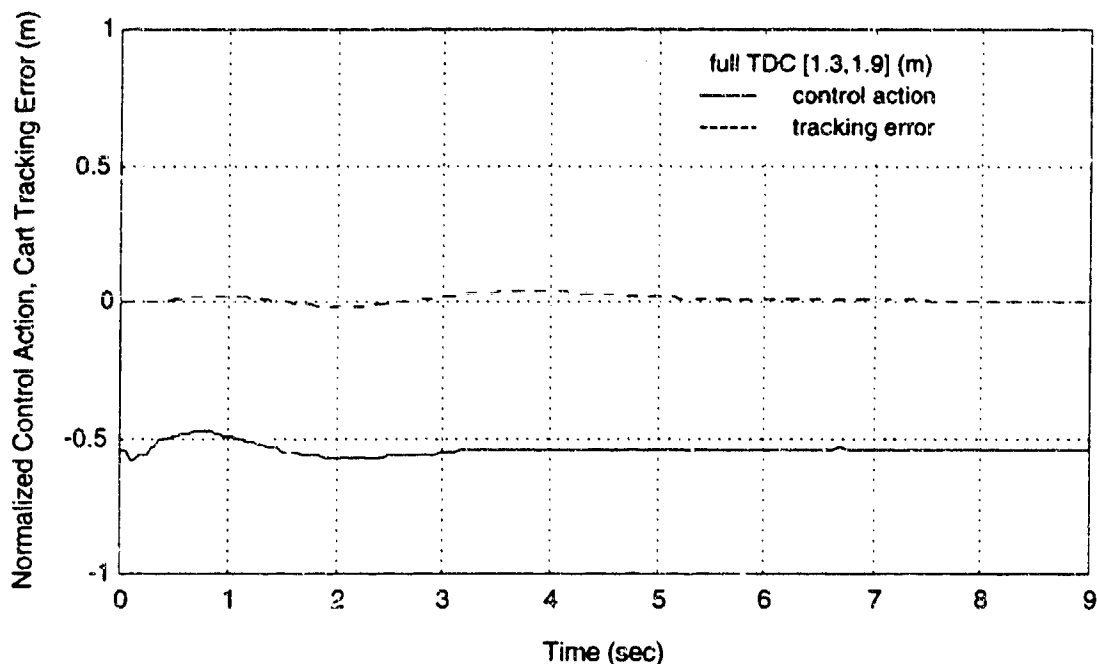


Figure 5.27 Full TDC, from 1.3 to 1.9 meters; force and position error

5.6 NOISE AND NONLINEAR FUNCTIONS OF CONTROL

The preceding three sections showed systematic testing of the two best hybrid architectures found. This section, for the sake of comparison, shows one run with a worse hybrid architecture, and one run with the best architectures in an unreasonably noisy environment.

Figures 5.28, 5.29, and 5.30 show the results for the hybrid controllers in an unreasonably noisy environment. On each time step, zero-mean, Gaussian noise was added to each sensor reading. For each element of the state vector, the noise had a variance equal to 10% of the total range that the element normally varies over, while following that trajectory. In practice, if an actual system had sensors this noisy, they would be filtered by a separate algorithm. Nevertheless, it is interesting to note that the hybrid is relatively insensitive to noise, and that it still performs well.

ATTACHMENT 1

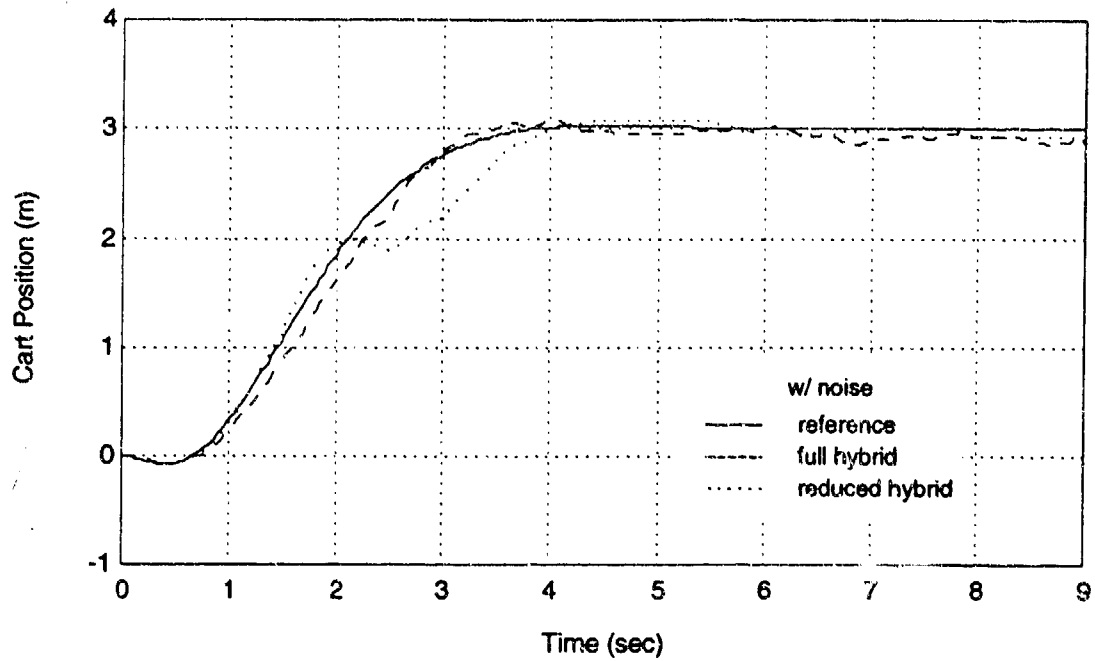


Figure 5.28 Full and reduced hybrid systems with 10% variance noise; cart position plot

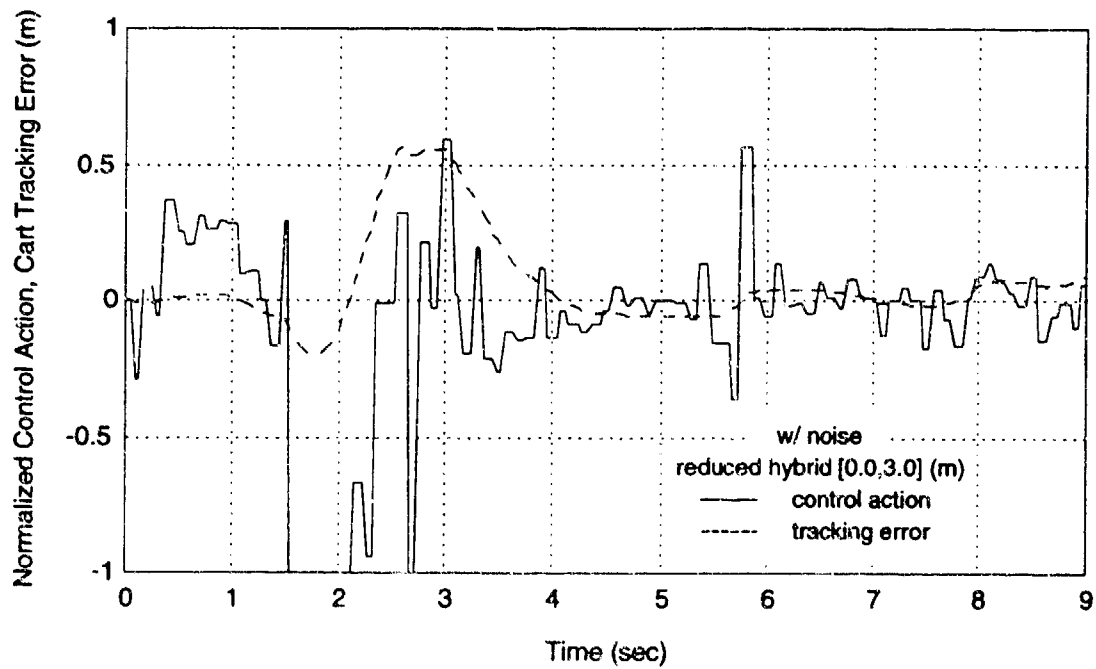


Figure 5.29 Reduced hybrid with 10% variance noise; force and position error

ATTACHMENT 1

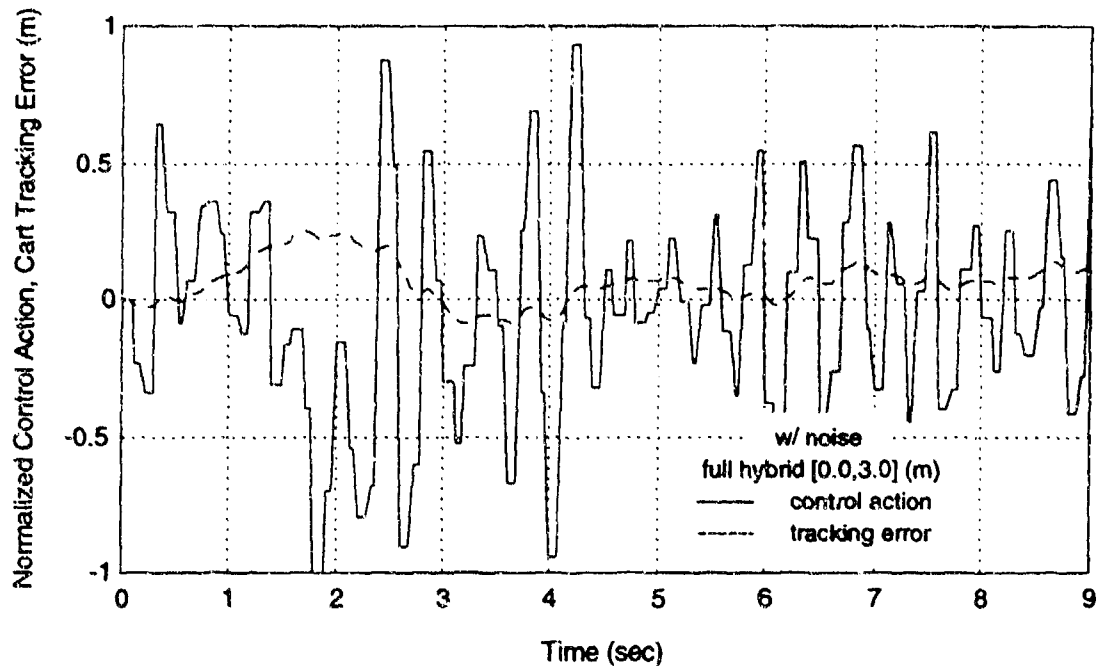


Figure 5.30 Full hybrid with 10% variance noise; force and position error

As can be seen from figure 5.28, both hybrid systems did extremely well. The full hybrid was slightly better than the reduced hybrid, but needed to apply more force. The performance difference was probably mainly do to the fact that the actuator saturated for a longer period in the case of the reduced controller, so that it was not able to apply as much force as it calculated was actually needed.

When the algorithm for the full hybrid was first developed, the network was allowed to learn a general, nonlinear function of both x and u . The results of using such a network are shown in figures 5.31 and 5.32.

ATTACHMENT 1

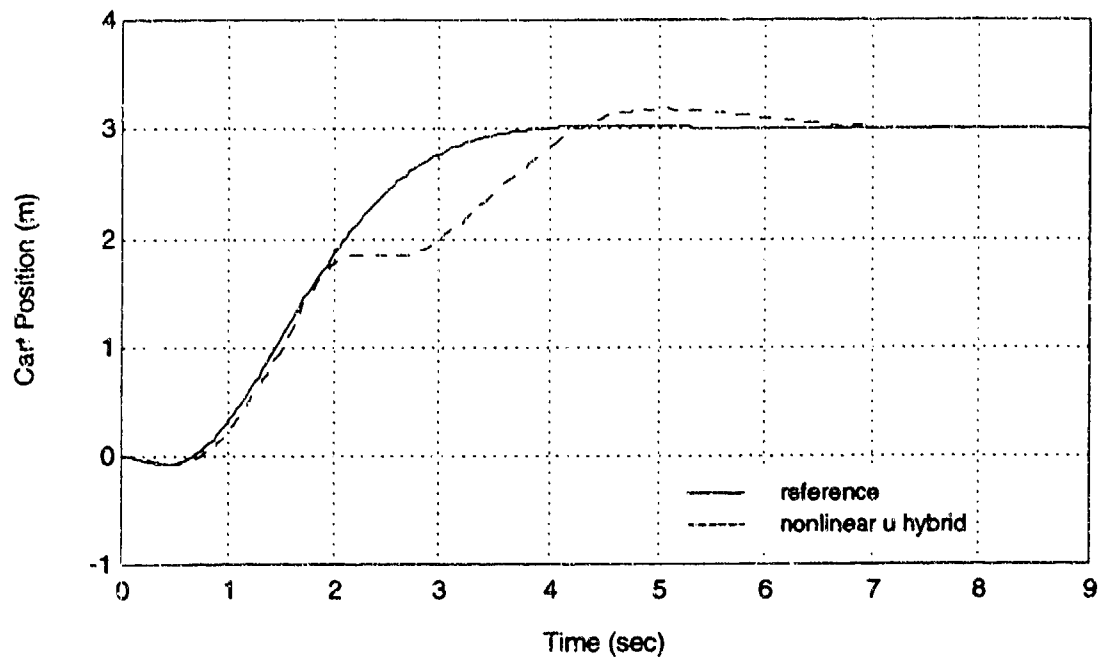


Figure 5.31 Full hybrid; general nonlinear function of both x and u ; cart position plot

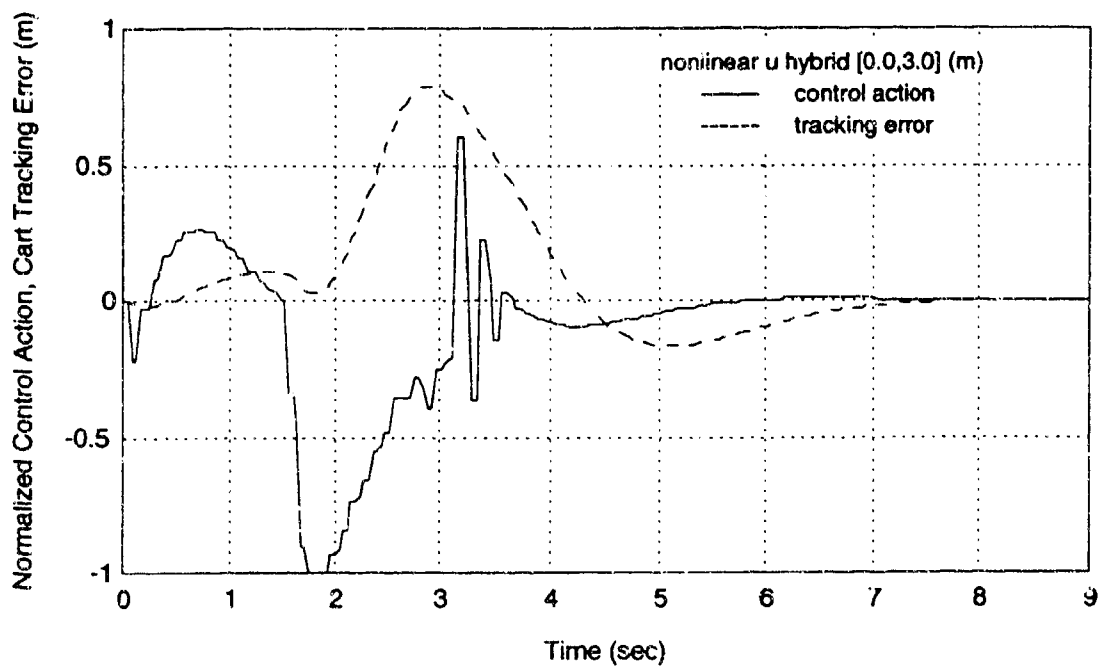


Figure 5.32 Full hybrid; general nonlinear function of both x and u ; force, position error

Although the pole never fell, the controller did not follow the reference path very closely. This controller was actually worse than TDC by itself. The problem arises

ATTACHMENT 1

because control action is one of the inputs to the network. In general, to find the correct control action to achieve the desired state, it is necessary to find the inverse (with respect to u) of the function implemented by the network. This can be a difficult problem. Nevertheless, because the unmodeled dynamics may often be considered as a fairly linear function of control, it should be possible to approximate the function in a given state as a linear function of control action. In other words, taking into account the unmodeled dynamics associated with the control action on the last time step, and assuming the partial derivative of Ψ with respect to u has not changed much, it should be possible to calculate the appropriate u for the current time step. When this idea was implemented, however, it did not make any significant difference. This may have been because the network actually learned Ψ as a nonlinear function of u . If a function is almost, but not quite, a line, then even if the distance between the function and the line is small everywhere, the difference between their slopes may be large. Learning the nonlinear function and then using its slope at some point evidently did not give enough new information to help much. A better approach would be to have the network learn the best *linear* function of u , and then look at the partial derivatives of this linear function. Of course, $\Psi(x,u)$ could still be a nonlinear function of x , and would only be constrained to be a linear function of u . Accordingly, a network was set up to learn Ψ as a possibly nonlinear function of x and a linear function of u . In fact, this alternative arrangement was used for the full hybrid experiment shown in figure 5.7, and can be seen to be significantly better than the hybrid used in figure 5.31.

Sections 5.1 through 5.6 have explored several different approaches to combining learned information with an adaptive controller. Using input/output partial derivative information in the control law seemed to be helpful, but only if the network was constrained to learn functions nonlinear in x and linear in u . Using the reduced canonical form had the advantage of allowing the network to learn a function with one output instead of four, and worked well enough that the partial derivatives were not needed. This system worked better for scenarios with computation delay and actuator dynamics, and worked

ATTACHMENT 1

equally well in the presence of noise. Overall, the full hybrid using partial derivatives tended to be the most effective controller, especially when the trajectory of the plant was largely in the region of greatest unmodeled dynamics.

5.7 COMPARISON OF CONNECTIONIST NETWORKS USED

5.7.1 Sigmoid

The network used in most of the above experiments was a Backpropagation, two layer, sigmoid network. Each of the inputs and outputs of the network were scaled before entering and after leaving it, so that each signal would vary over a range of unit width. Thus, the network would give equal preference to errors in each output. After trying several different learning rates, it was found that a rate of 0.005 worked best. The following graph (figure 5.33) shows the learning curve for the network while learning the function $\Psi(x,u)$, where Ψ was a nonlinear function of both x and u . The network output Ψ is a four element vector with one element for each of the four elements of state. The graph shows the base 10 logarithm of the error in the network's output, as a function of the training cycle.

ATTACHMENT 1

Semilog Plot of Error During Learning

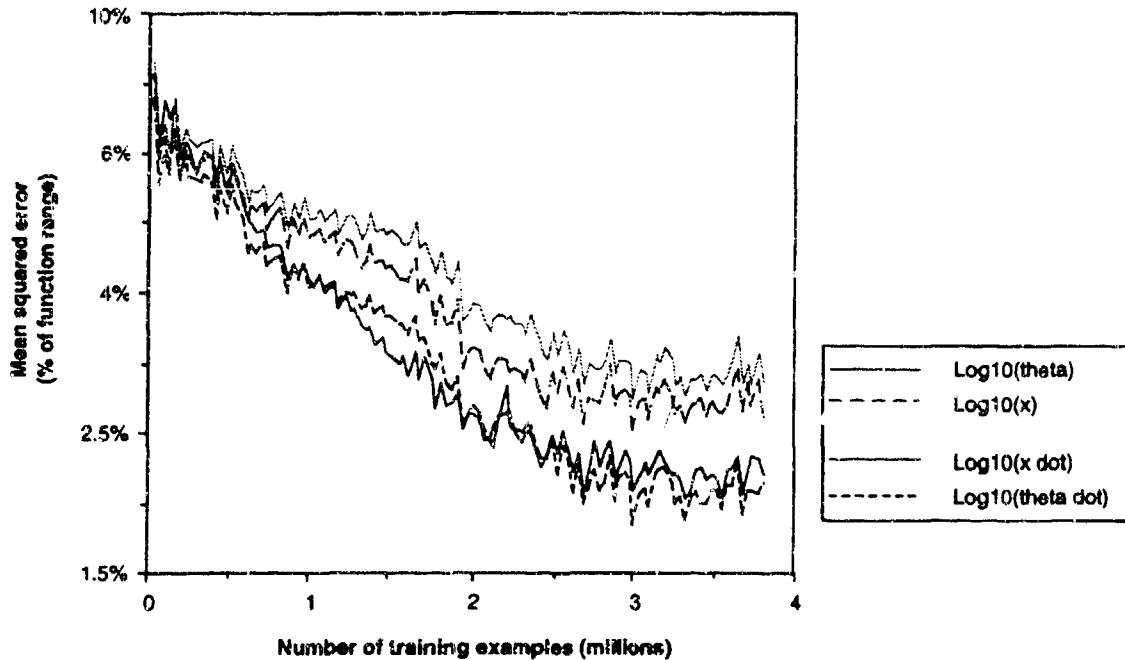


Figure 5.33 Learning curve

Even though each point in the curve is the average error in the output over a period of 400 training points, the curve still appears to be very noisy. This noise tends to cause the network to forget what it has learned unless the learning rate is fairly low, and so this noise is probably the reason that a learning rate of 0.005 was the largest rate that converged to a local minimum. Higher learning rates modified the weights so much on every step that they changed enough to forget previously learned information. Lower learning rates caused the network to learn even more slowly than in figure 5.33. The training period shown in the figure took approximately 63 hours to run on a Macintosh IIfx. Figure 5.34 shows a three-dimensional slice of the six-dimensional surface learned. In the figure, the three elements of state not shown are held at zero.

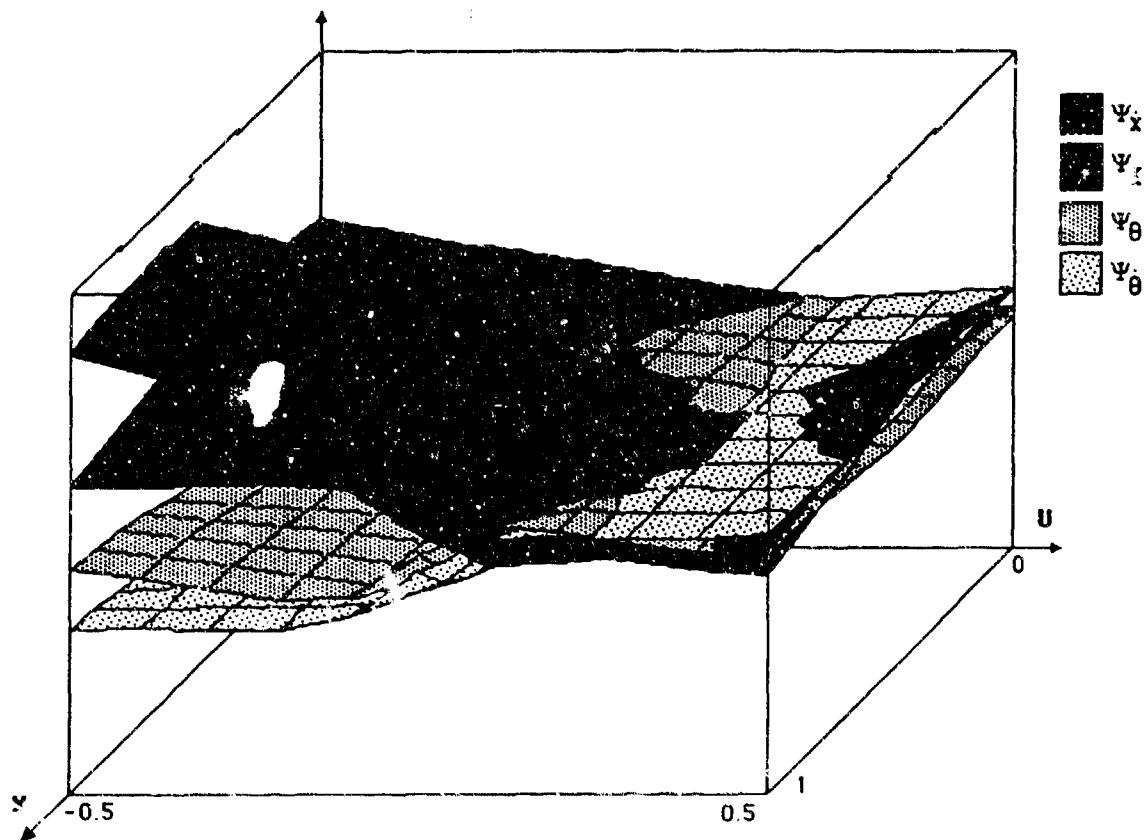


Figure 5.34 Function learned

The figure shows each element of Ψ as a separate surface, with all heights scaled to fit in the cube. The horizontal axis is the control action u , and the diagonal axis is the cart position x . The function is clearly nonlinear and widely varying in both of these dimensions, although it varies little along the other dimensions that are not shown.

As TDC generated new training points, these were stored in a buffer. The network was trained with points randomly drawn from this buffer. This was done to ensure that the network would not have problems with receiving a long string of training points all from the same region, thereby causing it to forget what it had already learned in other regions. Despite this random buffer, the network still learned extremely slowly.

A controller based on this approach would need one of three things to be practical. First, it could have special hardware to speed up the learning. Second, it might be in a situation where long learning times are acceptable. If a factory robot can learn to adjust to

ATTACHMENT 1

normal wear within a few days, then it should be able to learn any unmodeled dynamics due to wear faster than they occur. Third, the algorithm in the network might be modified to allow faster learning. This third approach was taken here.

5.7.2 Sigmoid With a Second-Order Method (Delta-Bar-Delta)

One attempt to speed learning was to apply a pseudo-Newton method to the sigmoid network. Delta-Bar-Delta [Jac91] was chosen because it requires very little extra computation time, and it has been compared favorably with a number of other methods. Unfortunately, comparisons between methods to speed learning are often done with benchmark problems that do not represent the problem here. People often compare learning speeds for learning an XOR function or a multiplexor function. These can be difficult problems for a network to learn, but the network has the advantage that the set of training points is finite and small, so it is not unreasonable to change weights only after each cycle through all the training data. Learning a function defined over a real vector is more difficult, since there is an infinite set of training points. The functions encountered in this thesis tended to be smooth and have few wrinkles, which meant that there was a large amount of redundancy in the data that the learning algorithm should have been able to exploit. These factors combined to yield a problem that was slow for Backpropagation alone to learn, but should have been learnable quickly by other learning methods.

When Delta-Bar-Delta was first applied, it immediately set all of the local learning rates to zero, causing the weights to freeze. This was because it worked by comparing the current partial derivative of error (with respect to a given weight) with an exponential average of recent values of this derivative. Since this was being done after every training point, it saw the noise in the training data and interpreted that as rapidly changing signs in the error derivatives. It responded to that by repeatedly decreasing all of the local learning rates.

The problem arose because Delta-Bar-Delta was not being used in an epoch

ATTACHMENT 1

training mode as it had been designed for. The apparent solution was to calculate two exponentially smoothed averages of the error partials. If these two averages had different time constants, then comparing them would be like comparing the current true derivative with a slightly older true derivative.

The values for these time constants were chosen heuristically. Looking at the learning curve for normal Backpropagation showed that the errors were noisy, but in a 500 training point period a "representative sample" of training points was probably being seen. The short-term average was therefore chosen so that 80% of the average was determined by the last 500 training points. The long-term average was then chosen to be 5 times slower, basing 80% of its value on the last 2500 training points. In normal Delta-Bar-Delta, the learning rate is increased by a constant every time the current derivative has the same sign as the long-term derivative average. Since this variation would update learning rates about 500 times more often, the rate of increase for learning rates was set 500 times smaller than is suggested for normal Delta-Bar-Delta. Similarly, when learning rates are decreased, the decrease is implemented exponentially by dividing by a constant each time. Since the modified Delta-Bar-Delta would be expected to divide by this constant 500 times as often, the 500th root of the suggested constant was used.

There are two novel ways that Delta-Bar-Delta can fail. If local learning rates are increased too often, then they get very large, and weights in the network can start to blow up. On the other hand, if local learning rates are decreased too often, then they rapidly approach zero, and the weights freeze. If the local learning rates stay within a reasonable range, then Delta-Bar-Delta can succeed or fail in the same manner as Backpropagation, although hopefully it reaches the final state faster.

In experimenting with Delta-Bar-Delta, every run either had exploding weights or vanishing learning rates. Given the very noisy training data that the network was exposed to, I was unable to find a useful set of parameters for Delta-Bar-Delta. It is, of course, possible that such a set of parameters exists. Perhaps Delta-Bar-Delta would work better if

ATTACHMENT 1

all the local learning rates were normalized on each time step to keep a constant average value, or perhaps some other heuristic might be applied. It is not immediately clear what would be the best way to deal with this problem.

6 CONCLUSIONS AND RECOMMENDATIONS

6.1 SUMMARY AND CONCLUSIONS

This thesis has described a new method for integrating an indirect adaptive controller with a learning system to form a hybrid controller, incorporating the advantages of each system. When a learning system is trained with the estimates found by the adaptive controller, the hybrid system reacts more quickly to unmodeled spatial dependencies in the plant. This hybrid system follows a reference trajectory better than the adaptive controller alone, but it can still be improved upon. By using a connectionist system to learn the function, it is easy to calculate the partial derivatives of that function, which in turn allows better estimates of unmodeled dynamics, and better estimates of the effect of control action on state. This modified controller performed better than either the adaptive controller alone or the original hybrid system.

The feedforward, sigmoid learning system was able to learn the required functions accurately, but the learning tended to be slow. The problem of slow convergence is widely recognized and is dealt with by methods such as Delta-Bar-Delta, which accelerate learning a great deal in published experiments. Unfortunately, those problems used for comparison usually involve small sets of training examples. The learning problem that arose in this thesis theoretically required an infinite training set. In practice, Delta-Bar-Delta was found to be very sensitive to the choice of learning parameters. Even modifying Delta-Bar-Delta to use two traces instead of one did not solve this problem, and it actually introduced another parameter that had to be chosen. Therefore, methods for accelerating convergence on small test problems do not appear to scale as well as commonly thought.

6.2 RECOMMENDATIONS FOR FUTURE WORK

The desired reference trajectory used in these experiments was chosen manually to give fairly fast response while still being achievable with the 10 Newton force constraint on the controller. It would be desirable to automate the choice of reference, and this may be possible. The reference trajectory could start off as a poor controller which is achievable without using much force. It could then be slowly improved (automatically) until the actuators nearly saturate, thus finding the best reference that can be matched by this hybrid controller architecture. The reference could even be a function of state, stored in a separate connectionist network.

The learning systems used here learned very good approximations, but the learning tended to be slow. The Delta-Bar-Delta algorithm improves the rate of convergence for small sets of training points, but was not effective for learning as part of a hybrid control system, even after being modified. It tended to be too sensitive to the choice of learning parameters. Learning based on following the first derivative should be faster if accurate measurements of the second derivatives can be found, so a system such as Delta-Bar-Delta should be useful if it can automate the choice of parameters, perhaps based on an estimate of how accurate its second derivative estimates are. Further research should focus on this problem, perhaps by measuring the standard deviation of the individual measurements to form an estimate of the accuracy of their average value.

BIBLIOGRAPHY

- [Åst83] Åström, K., "Theory and Application of Adaptive Control - A Survey," *Automatica*, Vol. 19, No. 5, 1983.
- [BB90] Baird, L. and W. Baker, "A Connectionist Learning System for Nonlinear Control," *Proceedings, AIAA Conference on Guidance, Navigation, and Control*, Portland, OR, August, 1990.
- [BF90] Baker, W. and J. Farrell, "Connectionist Learning Systems for Control," *Proceedings, SPIE OE/Boston '90*, (invited paper), November, 1990.
- [Bar89] Barto, A., "Connectionist Learning for Control: An Overview," COINS Technical Report 89-89, Department of Computer and Information Science, University of Massachusetts, Amherst, September, 1989.
- [BS90] Barto, A., and S. Singh, "Reinforcement Learning and Dynamic Programming," *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CN, August, 1990.
- [BSA83] Barto, A., R. Sutton, and C. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, No. 5, September/October 1983.
- [BSW89] Barto, A., R. Sutton, and C. Watkins, "Learning and Sequential Decision Making," COINS Technical Report 89-95, Department of Computer and Information Science, University of Massachusetts, Amherst, September, 1989.
- [D'A88] D'Azzo, J., *Linear Control System Analysis & Design: Conventional and Modern*, McGraw-Hill, New-York, 1988.
- [FGG90] Farrell, J., Goldenthal, W., and K. Govindarajan, "Connectionist Learning Control Systems: Submarine Heading Control," *Proceedings, 29th IEEE Conference on Decision and Control*, December, 1990.
- [Fu86] Fu, K., "Learning Control Systems - Review and Outlook," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, No. 3, May, 1986.
- [GF90] Goldenthal, W. and J. Farrell, "Application of Neural Networks to Automatic Control," *Proceedings, AIAA Conference on Guidance, Navigation, and Control*, August, 1990.
- [HW89] Hornik, K., and M. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, 1989.

ATTACHMENT 1

- [Jac91] Jacobs, R., "Increased Rates of Convergence Through Learning Rate Adaptation," *Neural Networks*, 1,2, p. 295-301, 1991.
- [Jam90] Jameson, J., "A Neurocomputer Based on Model Feedback and the Adaptive Heuristic Critic," *Proceedings of the International Joint Conference on Neural Networks*, 1990.
- [Jor88] Jordan, M., "Supervised Learning and Systems with Excess Degrees of Freedom," Technical Report COINS TR 88-27, Massachusetts Institute of Technology, 1988.
- [Klo88] Klopff, H., "A Neuronal Model of Classical Conditioning," *Psychobiology*, vol. 16 (2), 85-125, 1988.
- [LeC87] LeCun, "Modeles Connexionnistes de l'Apprentissage," Ph.D Thesis, Universite Pierre et Marie Curie, Paris, 1987.
- [MC68] Michie, D., and R. Chambers, "Boxes: an Experiment in Adaptive Control," *Machine Intelligence*, vol. 2, E. Dale and D. Michie, Eds., Edinburgh, Scotland: Oliver and Boyd Ltd., 1968.
- [Mil91] Millington, P., "Associative Reinforcement Learning for Optimal Control," Master's Thesis, Massachusetts Institute of Technology, 1991.
- [MP69] Minsky, L. and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.
- [NW89] Nguyen, D., and B. Widrow, "The Truck Backer-Upper: An Example of Self-Learning in Neural Networks," *Proceedings of the International Joint Conference on Neural Networks*, 1989.
- [Pal83] Palm, W., *Modeling, Analysis, and Control of Dynamic Systems*, John Wiley & Sons, New York, 1983.
- [Par82] Parker, D., "Learning Logic," Invention Report, S81-64, File 1, Office of Technology Licensing, Stanford University, 1982.
- [Ros62] Rosenblatt, F., *Principles of Neurodynamics*, Spartan Books, Washington, 1962.
- [RZ86] Rumelhart, D. and D. Zipser, "Feature Discovery by Competitive Learning," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, Rumelhart, D., and J. McClelland, ed., MIT Press, Cambridge, MA, 1986.
- [RHW86] Rumelhart, D., G. Hinton, and R. Williams, "Learning Internal Representation by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, Rumelhart, D., and J. McClelland, ed., MIT Press, Cambridge, MA, 1986.
- [Sam67] Samuel, A., "Some Studies in Machine Learning Using the Game of Checkers II - Recent Progress," *IBM Journal of Research and Development*, 11,601-617, 1967.

ATTACHMENT 1

- [Sam59] Samuel, A., "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, 3, 210-229, 1959, reprinted in *Computers and Thought*, A. Feigenbaum and J Feldman, ed., McGraw-Hill, New York, 1959.
- [Sim87] Simpson, P., "A Survey of Artificial Neural Systems", " Technical Document 1106, Naval Ocean Systems Center. San Diego, CA, 1987.
- [Sut90] Sutton, R., "Artificial Intelligence by Approximating Dynamic Programming," *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CN, August, 1990.
- [Sut88] Sutton, R., "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, Kluwer Academic Publishers, Boston, MA, vol. 3: 9-44, 1988.
- [Wat89] Watkins, C., "Learning from Delayed Rewards," Ph.D. thesis, Cambridge University, Cambridge, England, 1989.
- [Wer89] Werbos, P., "Backpropagation and Neurocontrol: A Preview and Prospectus," *Proceedings of the International Joint Conference on Neural Networks*, Washington, D.C., pp. 209-216, vol. I, 1989.
- [Wer74] Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD Dissertation, Harvard University, 1974.
- [Wid89] Widrow, B., "ADALINE and MADALINE," *Proceedings of the International Joint Conference on Neural Networks*, 1989.
- [WZ89] Williams, J. and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, 1, 270-280, 1989.
- [WB90] Williams, R. and L. Baird, "A Mathematical Analysis of Actor-Critic Architectures for Learning Optimal Controls Through Incremental Dynamic Programming," *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CN, August, 1990.
- [Wil88] Williams, R., "Towards a Theory of Reinforcement Learning Connectionist Systems," Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, July, 1988.
- [YI90] Youcef-Toumi, K. and O. Ito, "A Time Delay Controller for Systems with Unknown Dynamics," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 112, March, 1990.

ATTACHMENT 2

Reprint of:

Nistler, N. (1992). *A Learning Enhanced Flight Control System for High Performance Aircraft*, CSDL Report T-1127, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T.

A LEARNING ENHANCED FLIGHT CONTROL SYSTEM FOR HIGH PERFORMANCE AIRCRAFT

by

Noel F. Nistler

Submitted to the Department of Aeronautics and Astronautics
on May 8, 1992 in partial fulfillment of the requirements for the
Degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

Numerous approaches to flight control system design have been proposed in an attempt to govern the complex behavior of high performance aircraft. Gain scheduled linear control and adaptive control have traditionally been the most widely used methodologies, but they are not without their limitations. Gain scheduling requires large amounts of *a priori* design information and costly manual tuning in conjunction with flight tests, while still lacking an ability to accommodate unmodeled dynamics and model uncertainty beyond a limited amount of robustness that can be incorporated into the design. Adaptive control is suitable for nonlinear systems with unmodeled dynamics, but has deficiencies in accounting for quasi-static state dependencies. Moreover, inherent time delays in adaptive control make it difficult to match the performance of a well-designed gain scheduled controller. An alternative approach that is able to compensate for the inadequacies experienced with traditional control techniques and to automate the tuning process is desired.

Recent learning techniques have demonstrated an ability to synthesize multivariable mappings and are thus able to learn a functional approximation of the initially unknown state dependent dynamic behavior of the vehicle. By combining a learning component with an adaptive controller, a new hybrid control system that is able to adapt to unmodeled dynamics and novel situations, as well as to learn to anticipate quasi-static state dependencies is formed.

This thesis explores the concept of augmenting an adaptive flight controller with a learning system. The goal is to examine the extent to which learning can be used to improve the performance of an adaptive flight control system architecture, as well as to highlight some of the difficulties introduced by learning augmentation. Performance of the control system is defined in terms of its ability to control a nonlinear, three-degree-of-freedom aircraft model reacting to altitude and velocity commands. This hybrid approach offers potential advantages over conventional techniques in terms of performance, model uncertainty accommodation, and tuning costs.

Thesis Supervisor: Dr. Milton B. Adams
Title: Lecturer, Department of Aeronautics and Astronautics

Technical Supervisor: Mr. Walter L. Baker
Title: Senior Member of Technical Staff, Draper Laboratory

ATTACHMENT 2

ACKNOWLEDGMENT

This thesis was prepared at The Charles Stark Draper Laboratory, Inc. with support provided by the U.S. Air Force Wright Laboratory under Contract F33615-88-C-1740. Publication of this report does not constitute approval by Draper Laboratory or the sponsoring agency of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.

ATTACHMENT 2

TABLE OF CONTENTS

| | | |
|-------|--|-----|
| 1 | INTRODUCTION | 251 |
| 1.1 | Problem Description | 252 |
| 1.2 | Thesis Objectives | 253 |
| 1.3 | Overview | 253 |
| 2 | BACKGROUND | 255 |
| 2.1 | High Performance Aircraft Characteristics | 255 |
| 2.2 | Traditional Control Techniques | 257 |
| 2.2.1 | Robust Control | 258 |
| 2.2.2 | Gain Scheduling | 258 |
| 2.2.3 | Adaptive Control | 259 |
| 2.3 | Connectionist Learning Systems | 260 |
| 2.3.1 | Foundations of Connectionist Systems | 261 |
| 2.3.2 | Early Connectionist Networks | 263 |
| 2.3.3 | The Backpropagation Network | 265 |
| 2.3.4 | Connectionist Learning Systems for Control | 269 |
| 3 | TECHNICAL APPROACH | 272 |
| 3.1 | Adaptive Control Component | 272 |
| 3.1.1 | Control Law Derivation | 274 |
| 3.1.2 | Implementation Issues | 278 |
| 3.2 | Learning Control System | 279 |
| 3.2.1 | Incremental Learning and Fixation | 279 |
| 3.2.2 | Spatially Localized Learning | 281 |
| 3.2.3 | The Linear-Gaussian Network | 282 |

ATTACHMENT 2

| | | |
|-------|--|-----|
| 3.3 | Hybrid Learning / Adaptive Control | 288 |
| 3.3.1 | Hybrid Control System Architecture | 288 |
| 3.3.2 | Learning Versus Adaptation | 289 |
| 3.3.3 | Control Law Development | 291 |
| 4 | EXPERIMENTS | 296 |
| 4.1 | Aeroelastic Oscillator | 297 |
| 4.1.1 | Description | 297 |
| 4.1.2 | Open Loop Dynamics | 299 |
| 4.1.3 | Reference Model | 299 |
| 4.1.4 | Application of Hybrid Controller | 300 |
| 4.1.5 | Aeroelastic Oscillator Experiment 1 | 304 |
| 4.1.6 | Aeroelastic Oscillator Experiment 2 | 311 |
| 4.2 | High Performance Aircraft Model | 315 |
| 4.2.1 | Aircraft Description | 315 |
| 4.2.2 | General Aircraft Characteristics | 320 |
| 4.2.3 | Aircraft Reference Model | 324 |
| 4.2.4 | Application Issues | 325 |
| 4.2.5 | High Performance Aircraft Experiment 1 | 327 |
| 4.2.6 | High Performance Aircraft Experiment 2 | 339 |
| 5 | CONCLUSIONS AND RECOMMENDATIONS | 345 |
| 5.1 | Summary and Conclusions | 345 |
| 5.2 | Recommendations for Future Work | 346 |
| | BIBLIOGRAPHY | 348 |

1 INTRODUCTION

Numerous approaches to flight control system design have been proposed in an attempt to govern the behavior of high performance aircraft. This class of aircraft presents formidable challenges to the designer since by nature their dynamics are nonlinear, multivariable, and coupled (Etkin (1982)). Moreover, high performance aircraft tend to exhibit modes with relatively high natural frequencies and minimal damping as compared to typical aircraft. Gain scheduled linear control and adaptive control appear to be the most popular methodologies for flight control law design, but they are not without their limitations. Gain scheduling techniques combine multiple linear control laws to formulate a nonlinear controller (Lewis & Stevens (1992)). This process requires large amounts of *a priori* model information and potentially costly manual tuning, since a separate linear controller must be designed for each of a selected set of distinct regions of the operating envelope. In addition to this tedious design approach, gain scheduled controllers lack the ability to accommodate unmodeled dynamics and model uncertainty beyond a limited amount of robustness that can be incorporated into the design. Adaptive control is suitable for nonlinear systems with unmodeled dynamics but has deficiencies in effectively accounting for quasi-static state dependencies. Moreover, inherent time delays of adaptive control make it difficult to match the performance of an ideal gain scheduled controller (Stein (1980)). This thesis presents an alternative approach that compensates for some of the inadequacies experienced with these traditional control techniques.

By combining an adaptive component with a learning system, an innovative new hybrid controller is formed that allows each mechanism to focus on the control objective for which it is best suited. The primary role of the adaptive control component in the hybrid system is to accommodate unmodeled dynamics (i.e., dynamical behavior that is not

ATTACHMENT 2

expected, based on the design model). Additionally, the adaptive component has the auxiliary task of providing estimates of any observed unmodeled state dependent dynamic behavior to the learning system (i.e., unknown dynamics that are a function of state in areas of the state space where learning has not occurred). These estimates are obtained by observing previous plant behavior, essentially providing delayed estimates. Moreover, since no use is made of past estimates, the adaptive component can be considered to act without memory. Based on the estimates from the adaptive component, a learning system can be used to learn a functional approximation of these state dependencies and ultimately reduce model uncertainty in the system. Connectionist networks (which include artificial neural networks) have demonstrated the ability to synthesize highly nonlinear, multivariable mappings (Funahashi (1988), Hornik, *et al.* (1989)). More specifically, spatially localized connectionist networks have been proposed as an appropriate learning system for control applications (Baker & Farrell (1992)). Armed with a mapping from the learning system that represents the previously unknown state dependencies, the hybrid controller can anticipate vehicle behavior that is a function of state and compensate accordingly, effectively removing the delay in the estimates provided by an adaptive controller. The impact of a controller that has the ability to anticipate vehicle behavior can be seen in improved closed-loop system performance. Moreover, this ability to learn state dependencies offers advantages over conventional techniques in terms of model uncertainty accommodation and automation of the tuning process.

1.1 PROBLEM DESCRIPTION

This thesis presents the development and application of a hybrid control system to the problem of flight control for a high performance aircraft. Time Delay Control (TDC), a model reference adaptive controller, is augmented by a linear-Gaussian connectionist network, to form the hybrid flight control system. This hybrid system is applied to the

ATTACHMENT 2

control of the longitudinal motion of a high performance aircraft during various altitude and velocity maneuvers. Due to nonlinearities, model uncertainty, unknown dynamics, and a host of other difficulties, high performance aircraft present a significant challenge to the development of flight control systems.

1.2 THESIS OBJECTIVES

This thesis explores the use of a learning system to augment an adaptive flight controller. The extent to which learning can be used to improve an adaptive flight control system architecture, as well as the difficulties introduced by learning augmentation, are examined. The primary objective of this thesis is to illustrate the advantages of a hybrid adaptive / learning control system in terms of its ability to accommodate unmodeled dynamics and reduce state dependent uncertainties in the system model. This hybrid approach offers advantages over conventional techniques in terms of performance, robustness, and design refinement costs.

1.3 OVERVIEW

In Chapter 2, the challenges associated with high performance aircraft control law design are outlined. Moreover, background information on traditional control techniques is provided to serve as a foundation for the hybrid control law development, and also as a basis for comparison of alternative designs. The theoretical concepts underlying connectionist learning systems, as well as some approaches in using learning systems for control, are also presented.

In Chapter 3, the technical aspects of the hybrid control law are developed. This is accomplished by first presenting the underlying theory of the adaptive component and the spatially localized learning system before moving on to the derivation of the hybrid system.

ATTACHMENT 2

General characteristics of the hybrid controller are also presented.

In Chapter 4, two experiments are presented to illustrate the implementation and performance of the hybrid control law. The first experiment uses the hybrid system to control a relatively simple nonlinear aeroelastic oscillator. Due to the low dimensionality of the plant, and a known truth model, the analysis and evaluation of the hybrid control system for the aeroelastic oscillator is greatly simplified. In the second experiment, the hybrid system is applied to a realistic high performance aircraft model. Descriptions of the major components of the aircraft model as well as its significant characteristics are also provided. An evaluation of aircraft performance when controlled by the hybrid system is presented and compared with other designs for various simulations. Learning system characteristics are also described.

Chapter 5 summarizes the major contributions of this thesis. In addition, recommendations for future research are presented.

A bibliography of the works used in preparing this thesis follows Chapter 5.

2 BACKGROUND

The design of automatic flight control systems for high performance aircraft presents significant challenges for the control engineer. Although well-developed design methodologies exist for linear systems, similar methodologies and related theories for nonlinear systems have proven to be elusive. In this chapter, the formidable challenges inherent in high performance aircraft control system design are presented in Section 2.1, conventional control approaches for accommodating these difficulties are presented in Section 2.2, while the fundamentals of connectionist learning systems and some approaches for learning control are introduced in Section 2.3.

2.1 HIGH PERFORMANCE AIRCRAFT CHARACTERISTICS

Because the aerodynamic forces and moments that act on an aircraft are complicated, nonlinear functions of many variables, aircraft exhibit complex flight dynamics. This section discusses the major difficulties associated with high performance aircraft flight control design.

Due to the high cost and dangers involved in flight testing, the majority of the effort in flight control system design and development relies on a model of the aircraft instead of the actual vehicle. This approach guarantees the presence of *model uncertainty* since it is impossible to capture the complete dynamical behavior of complex aircraft in a model. Errors in the model can be attributed to two major factors: structural and parametric uncertainty (Eker & Farrell (1991)). Typically, the mathematical structure of an aircraft model is derived from the general equations-of-motion for a single, rigid body. These are the classical Euler equations. From this base set of equations, the designer determines

ATTACHMENT 2

additional effects that must be included to obtain an effective flight control system design. Gyroscopic effects due to the presence of spinning rotors and aeroelastic effects due to inaccuracies in the rigid body assumption have historically been incorporated into the equations-of-motion. Beyond the difficulties associated with the selection and development of the proper model structure, the accuracy of the actual parameter values used in the model plays a large role in the quality of an aircraft model. Since values of the parameters are typically obtained from wind-tunnel testing or computational fluid dynamics (e.g., computer simulations of airflow over an aircraft model), large discrepancies are possible. Additional model uncertainty develops from the fact that not all flight conditions can be easily modeled by a single global model structure. For this reason, separate models are needed for post-stall flight, vertical take-off modes, and other extreme flight conditions. In general, all models contain a degree of uncertainty that must be addressed by the flight control system.

Nonlinearities present a major difficulty to the control engineer since no general theory for control design synthesis has been developed for nonlinear systems. Aircraft dynamical behavior is inherently nonlinear; this nonlinear behavior is caused primarily by the fact that the aerodynamic forces and moments that dictate aircraft motion are themselves complicated, nonlinear functions of many variables. Moreover, the full six-degree-of-freedom rigid body equations-of-motion include nonlinear terms. The effects of actuator rate limiting, control position limits, and other control linkages are further examples of nonlinearities.

Another complication experienced with flight control law design is that high performance aircraft are inherently *high dimensional, multivariable* systems. A six-degree-of-freedom aircraft requires twelve coupled state equations to fully characterize its rigid body dynamics. Moreover, multiple control effectors (e.g., stabilator, rudder, ailerons, and throttle) are employed to achieve the primary objective of simultaneously controlling a number of outputs (e.g., altitude, heading, and velocity). As a result, any

ATTACHMENT 2

control system that attempts to decouple the dynamics and connect independently designed single-input / single-output controllers will generally sacrifice performance for ease of design.

The "high performance" qualifier on the aircraft model implies expanded flight regimes that also tend to exacerbate control difficulties. These regimes include high angle-of-attack, high Mach, and other regions of the aircraft envelope where large changes in the aircraft dynamics can be expected. For example, a dynamic mode that is stable and adequately damped in one region of the envelope may become lightly damped or unstable in another. This fact, combined with the general trend toward relaxed static stability, requires rapid control action to stabilize the aircraft.

The above discussion illustrates the major challenges in flight control law design. Additional difficulties confront the control engineer due to the design methods themselves (e.g., frequency domain methods do not easily lend themselves to multivariable control) and due to challenges in applying the control approach to the real vehicle (e.g., digital implementation issues).

2.2 TRADITIONAL CONTROL TECHNIQUES

Automatic flight control systems have evolved from the "Sperry Aeroplane Stabilizer," the first functional autopilot, to advanced multivariable digital systems capable of generating a large number of control actions per second (Lewis & Stevens (1992)). Of the multitude of design theories and methodologies developed for flight control law design, the majority can be classified into the two broad categories: fixed control (e.g., robust control and gain scheduled control) and adaptive control. The following sections introduce these traditional control approaches. Each technique is critiqued in its ability to accommodate the design difficulties presented in the previous section.

2.2.1 Robust Control

Robust control has gained popularity for flight control due to its ability to accommodate a certain degree of uncertainty associated with the aircraft model. By explicitly incorporating uncertainty into the design process, robust controllers provide performance and stability guarantees. However, this resilience to uncertainty, or robustness, is usually obtained at the expense of a loss in system performance. Since typical robust control techniques (e.g., classical Bode gain / phase margin methods or H_∞ design) rely on a worst case estimate of the modeling error or margins to determine a fixed parameter control system, the resulting control law is often conservative when applied to the nominal plant and presents a tradeoff between stability robustness and high performance. Thus, a control system designed to account for modeling uncertainty results in suboptimal performance relative to the ideal case where no model uncertainty exists. To increase performance, the designer can exploit an improved model having less uncertainty. However, the added complexity and cost of a more refined model often prohibits this course of action. Beyond difficulties in achieving maximum performance, robust controllers are ill-adapted to handle highly nonlinear systems or unmodeled dynamics. In particular, although slight perturbations due to nonlinearities or unknown dynamics can be accommodated by further increasing the bounds on uncertainty, difficulties in achieving adequate performance are further exacerbated. For highly nonlinear aircraft with substantial unmodeled dynamics or model uncertainty, robust control is impractical from a performance point of view.

2.2.2 Gain Scheduling

Flight control systems for modern high performance aircraft are generally developed with a gain scheduling design methodology. Gain scheduling methods combine multiple linear control laws to formulate a nonlinear controller. This control approach can accommodate many of the difficulties associated with complex nonlinear systems, such as

high performance aircraft. To formulate this nonlinear control law, the operating envelope is separated into an *ad hoc* set of distinct regions where the dynamical behavior is approximately linear. By linearizing the dynamics in each distinct region, the designer is able to utilize the large class of linear control theories (e.g., robust or optimal approaches) to develop a control law best suited to realize local performance objectives. The combined nonlinear control law is achieved by transitioning among these linear control laws as flight conditions move among the prescribed linearized regions. Transitioning is accomplished by interpolating the control parameters (e.g., feedback gains) as a function of scheduling variables or operating condition. Mach number, angle-of-attack, and dynamic pressure are the most commonly used scheduling variables. As a result, highly nonlinear systems require numerous linearized regions, and subsequently a multitude of linear control laws, to approximate nonlinear behavior.

In addition to the subjective (and tedious) nature of defining a set of linearized operating regimes and designing a linear control law for each linearization point, gain scheduled flight control systems are also susceptible to model uncertainty and unmodeled dynamics. Differences between the observed and predicted vehicle behavior can only be corrected by on-line manual tuning during flight testing.

2.2.3 Adaptive Control

Adaptive control has been suggested as a viable method for aircraft flight control (Lewis & Stevens (1992), Stein (1980)). Adaptive techniques generally rely on differences between desired and observed vehicle behavior to adjust (adapt) variable internal parameters to ultimately achieve acceptable closed-loop performance. Using this approach, adaptive controllers have shown an ability to accommodate nonlinear plants with unmodeled dynamics. However, adaptive controllers encounter difficulties in systems with rapidly varying parameters and extensive nonlinearity. In an adaptive technique, the controller must wait until undesired plant behavior is observed before it can determine how

ATTACHMENT 2

to adjust its parameters. Potentially, several control intervals might be required to accurately detect and compensate for variations in these parameters. Beyond this delay associated with determining the correct parameters, sensor noise causes additional delay due to the required filtering. For vehicles that regularly experience large parameter variations, the resulting control law may spend large portions of time in some suboptimal, partially adapted configuration. This dilemma is exacerbated by the reactive nature of adaptive controllers in that the parameters must be re-tuned whenever the vehicle enters a new region, even if the correct values had previously been determined for that region. Hence, adaptive controllers fail to make use of predictable behavior (e.g., state dependencies) that would reduce the time spent in partially adapted states and ultimately improve performance. For these reasons, it is difficult for an adaptive controller to match the performance of a well designed gain scheduler control.

Although not as common as gain scheduling or adaptive approaches, multi-region adaptive controllers have also been suggested as a means for gain scheduling (Athans, *et al.* (1977), Stein, *et al.* (1977)). Essentially, each scheduler schedules multiple local plant models within an indirect adaptive control framework.

2.3 CONNECTIONIST LEARNING SYSTEMS

Connectionist learning systems have received much attention in the research community due to their potential in solving problems in pattern recognition, associative memory, and database retrieval (Rumelhart, *et al.*)). Moreover, recent attention has been given to exploring how connectionist networks can synthesize multivariable, nonlinear mappings and how this information can be applied to improve automatic control systems. In this section, a brief history of the development of a class of connectionist learning systems that is relevant to the control problem described earlier is presented. Some alternative approaches for incorporating connectionist systems into control system designs

are also introduced.

2.3.1 Foundations of Connectionist Systems

Connectionist systems, which include what are often called "artificial neural networks," owe their foundations to biologists and research psychologists who originally studied the ability of neural models to mimic the behavior of the brain (Rosenblatt (1962), Klopff (1988)). Contemporary connectionist systems have advanced significantly from these early beginnings (Barto, *et al.* (1983), Rumelhart, *et al.* (1986)). Many of the recent connectionist learning systems emphasize the mathematical theory of function approximation, estimation, and optimization (Baker & Farrell (1992), Poggio & Girosi (1990)).

Connectionist learning systems typically contain a large number of simple processing units that are combined in a highly interconnected architecture. These processing units, also known as nodes or "artificial neurons," make up the basic building blocks of a connectionist system. Figure 2.1 illustrates the internal structure of a simple, 3-input node

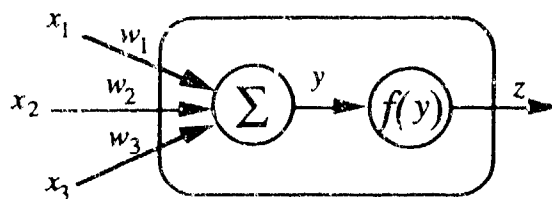


Figure 2.1 3-Input / 1-Output Simple Node

where x_1 , x_2 , and x_3 are the node inputs, w_1 , w_2 , and w_3 are weightings for the respective inputs, and y is the sum of the weighted inputs. The output of the network, z , is simply the value of the nodal function f evaluated at y . Nonlinear nodal functions are required to realize nonlinear mappings. Three examples of nodal functions are the threshold,

ATTACHMENT 2

sigmoidal, and Gaussian functions.

If a large amount of *a priori* information is known about the desired mapping of the network, the weights between the nodes can be set to fixed values to realize the network mapping. However, typical connectionist networks use nodes with fixed functions and adaptable weights that are adjusted using an appropriate learning law. Under supervised learning, the amount of weight adjustment is determined by evaluating an error formed by the difference between the calculated output of the network and a known desired output (Melsa (1989)). This contrasts with the weight adaptation by unsupervised learning, where only inputs and a reinforcement signal that characterizes past performance (i.e., not a known desired output) are utilized in adjusting the weights (Barto (1989), Mendel & McLaren (1970)). Thus, the operation of adaptable connectionist networks consists of two distinct phases: output calculation and learning. The output calculation phase is characterized by the determination of the network output based upon the given inputs, weights, and nodal functions. The purpose of the learning phase is to adjust the weights (using either a supervised or unsupervised technique) to obtain desired input / output behavior.

Connectionist networks are frequently categorized by the nodal architecture and associated output calculation or by the learning technique. One common architecture dependent on a specific output calculation method is the feedforward connectionist network (Funahashi (1988), Hornik, *et al.* (1989)). In feedforward structures, the output for any given node is not connected back as an input to itself by any feedback loop. Because of this feature, present outputs do not impact future output values (present outputs can impact future outputs in the learning phase by adjustment of the weights). Moreover, the output of the entire system can be calculated in a single pass since each layer simply outputs computed values based on inputs from the previous layer. Figure 2.2 illustrates a simple feedforward network.

ATTACHMENT 2

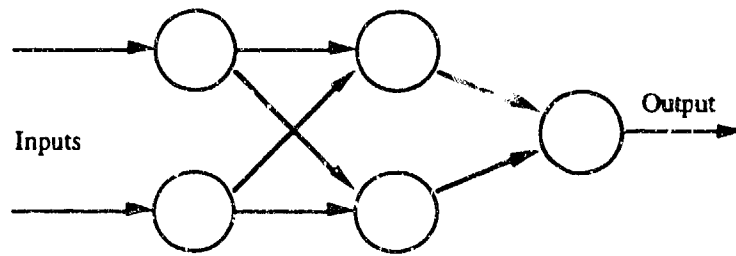


Figure 2.2 Simple 2-Input / 1-Output Feedforward Network

Another major class of connectionist systems consists of feedback (or recurrent) networks. The distinguishing feature of a feedback network is that nodes have the ability to influence themselves through feedback. The feedback can act directly from a given node to itself or indirectly through other nodes. Although feedback networks have an ability to learn dynamical mappings (e.g., mappings that change with time), the learning laws become complicated since the network output is no longer simply a function of network inputs and weights (it is also a function of the state of the network). Moreover, any feedback network representing a dynamical mapping can be expressed as an equivalent dynamic system of two static mappings separated by an integration or unit delay operator (Livstone, *et al.* (1992)).

By altering the nodal function, output calculation, learning approach, or a host of other variables, connectionist networks have been developed that display an array of different properties (Barto (1989), Melsa (1989), Minsky & Papert (1969)). Section 2.3.2 discusses some of the most popular early connectionist systems.

2.3.2 Early Connectionist Networks

One of the earliest uses of a connectionist methodology for learning was the *perceptron* network (Rosenblatt (1962)). A simple perceptron network is comprised of single or multiple layers of perceptron nodes connected in a feedforward configuration. A perceptron node is characterized by the binary threshold function used to formulate the output from the weighted sum of its inputs as shown in Figure 2.3. If the weighted sum is

ATTACHMENT 2

greater than some prescribed threshold value, the perceptron node outputs an "on" signal or the value 1. For inputs below the threshold, the node is considered "off" and outputs -1.

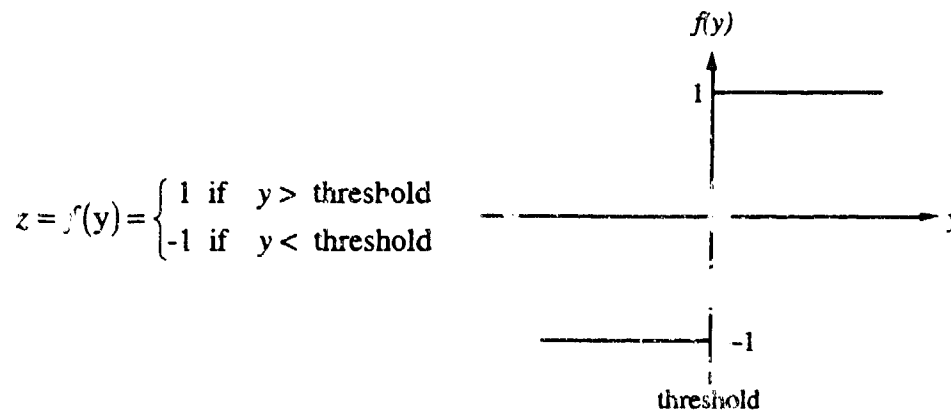


Figure 2.3 Binary Threshold Function

Perceptron networks have illustrated surprisingly powerful mapping capabilities. Minsky and Papert demonstrated the ability of single-layer perceptron networks to learn any discriminant function among classes that are linearly separable, using a simple learning rule (Minsky & Papert (1959)). The learning rule adjusts the weights incrementally depending on their impact on the error between the network output and the prescribed output. It was later shown that multi-layer perceptron networks are capable of discriminating a large class of nonlinearly separable problems. However, no general guarantee on the ability of any learning law to locate an optimal set of weights exists for multi-layer networks as in the single-layer case.

Another pioneering connectionist network is the adaptive linear element, or ADALINE (Widrow & Hoff (1960)). ADALINE networks consist of simple nodes connected in a feedforward architecture. The distinguishing features of an ADALINE network include a nodal function that simply outputs the weighted sum of the inputs (i.e., $f(y) = y$) and a normalized least mean square (LMS) learning law. Under supervised learning where the current inputs and desired output are known, the LMS learning law

ATTACHMENT 2

attempts to minimize the mean squared value of the error. When the weights are changed in proportion to the error, an ADALINE network is guaranteed to converge to the minimum of the mean squared error for linearly separable problems. In an attempt to extend this result to nonlinearly separable problems, ADALINES can be connected in a hierarchical structure to form a network of multiple adaptive linear elements (MADALINES). Although MADALINES are capable of producing complicated nonlinear mappings, determining the optimal weights between layers of ADALINES is a difficult process. These difficulties are the result of LMS learning laws being limited to the determination of optimal ADALINE weights and not the weights associated with their connecting layers (Melsa (1989)).

2.3.3 The Backpropagation Network

Although the advent of perceptrons, ADALINES, MADALINES, and their variants played a large role in the development of connectionist networks, the latest resurgence of interest in learning systems can be attributed to the backpropagation sigmoidal network. Although backpropagation is strictly speaking a learning law, its extensive use has resulted in the name being generalized to denote the large class of feedforward multi-layer networks that employ this particular learning approach. Similar to the early architectures, backpropagation networks are constructed from the combination of simple nodes arranged in a hierarchical, feedforward fashion. However, instead of the threshold and identity functions associated with the simple perceptron and ADALINE networks respectively, the backpropagation node uses a non-linear nodal function. One of the most commonly used nodal functions is the sigmoidal function illustrated in Fig 2.4.

ATTACHMENT 2

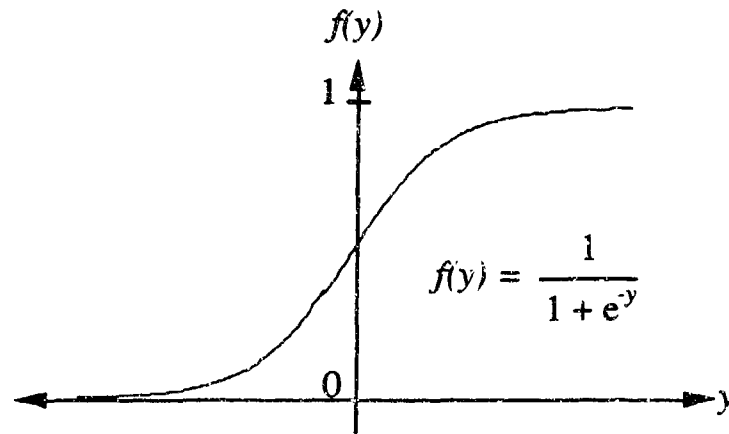


Figure 2.4 Sigmoidal Function

A sigmoidal function is a continuous, monotonically increasing function with finite asymptotic values. As a result, a sigmoid offers advantages over discontinuous nodal functions in that it is continuously differentiable, which plays a large role in the gradient learning algorithm described below.

A typical sigmoidal backpropagation network is shown in Figure 2.5. This network architecture is generally sub-divided into three distinct regions or layers: input layer, hidden layers, and output layer.

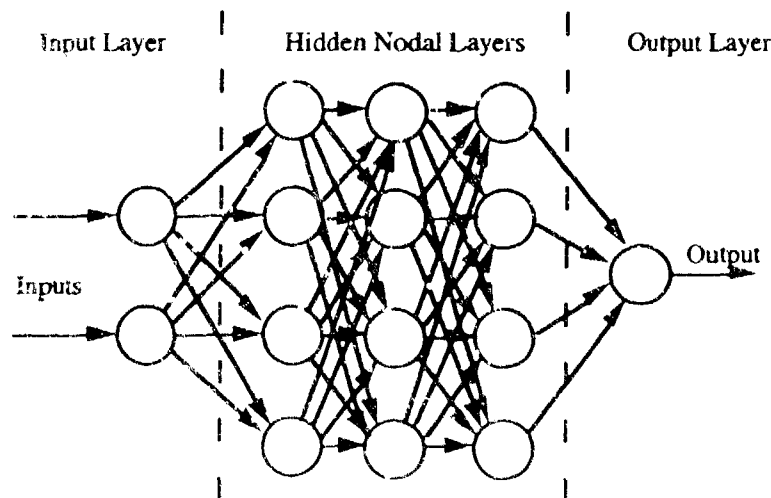


Figure 2.5 Typical 2-Input / 1-Output Feedforward Network with Three Hidden Layers

ATTACHMENT 2

The first region, the input layer, is characterized by nodes that act as an interface between network inputs and the subsequent hidden layer by simply passing the input value to a set of nodes in the first hidden layer (although weighting is sometimes added to the signal). Moreover, there is the same number of nodes in the input layer as there are inputs, and each input layer neuron typically passes its value to each node in the subsequent layer. The second region contains the hidden nodal layers. In this region, the weighted sum of the outputs from the previous layer is used as the input to each sigmoid function to compute the output of the node. The output is subsequently passed to a following hidden or output layer. The final region is the output layer, which contains the same number of output nodes as there are network outputs. The function of the output layer is to compute the weighted sum of its inputs and pass this value, or the value of a sigmoid function evaluated at this weighted sum, as the network output. Typically, the number of processing nodes in backpropagation networks is large compared to the number of different kinds of nodal functions used in the network, with networks using a single nodal function being the most common.

Although the selection of the network architecture is significant, the performance of connectionist networks is ultimately determined by the ability of the learning law to find the optimal weights. For backpropagation networks, weights are adjusted using an "error backpropagation" algorithm (Rumelhart, *et al.* (1986)). Whereas the learning laws of early connectionist networks had difficulties in properly adjusting connecting layer weights, the error backpropagation algorithm provides a systematic method to adjust weights in all adaptable layers. The basic error backpropagation algorithm uses a supervised gradient descent method to incrementally adjust the weights in the negative direction of the gradient (with respect to the weights) of a cost function. The general form of the gradient rule is shown in Equation (2.1) below:

$$\Delta w = -\alpha \frac{\partial J}{\partial w} \quad (2.1)$$

ATTACHMENT 2

where \mathbf{w} is a vector whose elements are the input weights, α is the learning rate (i.e., the step size), and J is the cost function to be minimized. The most commonly used cost function to be minimized is a quadratic function of the error between the network output and some desired output. In many supervised learning applications, the network is trained on a finite number of (known) input / output sample points. In this case, known as batch-mode training, the quadratic error cost function takes the following form.

$$J = \frac{1}{n} \sum_{i=1}^n [\mathbf{d}(\mathbf{x}_i) - \mathbf{f}_{net}(\mathbf{x}_i, \mathbf{w})]^T [\mathbf{d}(\mathbf{x}_i) - \mathbf{f}_{net}(\mathbf{x}_i, \mathbf{w})] \quad (2.2)$$

Here, n is the number of training examples, \mathbf{x}_i is the network input for the i^{th} training sample, $\mathbf{d}(\mathbf{x}_i)$ is the desired output at the i^{th} training sample, and $\mathbf{f}_{net}(\mathbf{x}_i, \mathbf{w})$ is the actual output of the network for the given input and weights. Using this technique, the weights are adjusted once per each pass or epoch through *all* the training examples. Recalling that the output of a layer is a function of the output of the previous layers, the partial derivatives of the cost function with respect to an individual weight can be found by forming a chain rule of partial derivatives and working backward along the same connections as the original forward path. Since the sigmoid is a continuous function, the partials always exist. Hence, propagation of the errors backward during the learning stage requires essentially the same amount of computation time as the forward calculation of the network output.

As with all gradient descent methods, the presence of local minima prevent any guarantees being placed on the ability of the learning algorithm to converge to the optimal solution. Moreover, simple gradient descent algorithms tend to converge slowly, especially if there are "troughs" in the error surface (Baird (1991)). Since the goal of learning is to follow the gradient in a downhill direction, a small learning rate results in slow convergence. If the learning rate is too high, the weight vector may completely bypass the trough to some possibly suboptimal plateau or oscillate across the bottom of the trough with little movement in the direction of the minimum.

If an acceptable learning rate is used, or if one of several techniques for speeding up

ATTACHMENT 2

convergence is applied (e.g., adding momentum terms to the weight update equation (Fujimelhart, *et al.* (1986)) or using second order derivative information on the cost (Jacobs (1988))), backpropagation networks have shown the ability to adequately map highly nonlinear functions. In fact, sigmoidal backpropagation networks with more than one hidden layer can represent any function to a desired degree of accuracy given enough nodes and training samples (Funahashi (1988), Hornik, *et al.* (1989)). This universal function approximation property has played a major role in the resurgence of the sigmoidal backpropagation network in applications ranging from pattern recognition to automatic control. However, one should recall that due to the presence of local minima in J , there is no guarantee that a given learning rule will actually yield the weights that represent the desired mapping.

Many variants of connectionist networks have been developed in an attempt at improved learning. However, the majority of all systems have one common characteristic. Learning is essentially a process of functional approximation, where inputs and desired outputs are synthesized to form a multivariable, nonlinear mapping. The type of learning system used and its associated details are dependent on the specific application. Section 3.2 presents one such specialized approach that is used for the learning augmented control of a high performance aircraft.

2.3.4 Connectionist Learning Systems for Control

Due to their ability to approximate smooth multivariable, nonlinear functions, connectionist learning systems have generated a large amount of interest among control engineers. However, a single, systematic approach for the application of connectionist learning systems to control has not yet materialized. This section briefly introduces a small subset of commonly used approaches for learning control, and lists references where further discussion may be found.

Copying an existing controller is perhaps one of the simplest techniques in

ATTACHMENT 2

applying connectionist learning systems to control. Assuming there exists a controller that is able to control the plant, the objective of the connectionist learning system is to synthesize the mapping between the inputs and the desired output supplied by the existing controller. Using this approach, the learning system can replace an existing controller in situations where the existing controller is impractical (e.g., where it is dangerous for a human to control the plant) or where the learning system offers a less costly representation. This approach was successfully applied to a pole balancing problem by Widrow & Smith (1964), where the existing control law was supplied by a human.

Direct inverse control is another method of applying a connectionist learning system to control (Werbos (1989)). Using this approach, the objective of the learning system is to identify the plant inverse. This is accomplished by providing the output of the plant as the network input and the input to the plant (i.e., control signals) as the desired network output. If the network has a plant inverse (i.e., if there is a unique plant input that produces a unique plant output), then when the desired plant output is provided as input to the network, the resulting network output is the control to be used as input to the plant (Barto (1989)). The drawbacks to this technique are that a desired reference trajectory must be known in order to supply the network with the desired plant output and the inverse of the plant must be well-defined (i.e., a 1-to-1 mapping between inputs and outputs must exist).

In the *backpropagation through time* method developed by Jordan (1988), two connectionist learning systems are used. The objective of the first network is to identify the plant, from which one can efficiently compute the derivative of the model output with respect to its input by means of back-propagation. Subsequently, propagating errors between actual and desired plant outputs back through this network produces an error in the control signal, which can be used to train the second network (Barto (1989)). This approach offers an improvement over direct inverse control since it is able to accommodate systems with ill-defined inverses, although the desired trajectory must still be known.

ATTACHMENT 2

Another approach for incorporating learning into a control system is to augment an adaptive technique with a learning system to form a hybrid controller (Baker & Farrell (1990), Baird (1991)). Augmentation of the adaptive technique may be implemented using a direct or indirect approach. Using a direct approach, the learning system generates a control action (or set of control parameters) associated with a particular operating condition. This control action is then combined with a control action produced by the adaptive system to arrive at the control that is applied to the plant. In contrast, for the indirect approach, the objective of the learning system is to improve the model of the plant. Here, the learning system generates model parameters that are a function of the operating condition. The learned model parameters are combined with adaptive estimates to arrive at a model of the plant. Given a presumably improved plant model, an on-line control law design is used to form the closed-loop system. A particular indirect learning augmented approach is used in this thesis and is developed in Chapter 3.

Reinforcement learning has also been suggested as a method of applying connectionist learning systems for control (Mendal & McLaren (1970), Barto (1989), Millington (1991)). The major difference between reinforcement learning and the previously discussed approaches is that under reinforcement learning, the objective is to optimize the overall behavior of the plant, so that no explicit reference / desired trajectory is required. As a result, reinforcement learning essentially involves two problems, the construction of a *critic* that is capable of evaluating plant performance in a manner that is consistent with the actual control objective, and the determination of how to alter controller outputs to improve performance as measured by the critic (Barto (1989)). The latter of the two problems can be addressed by one of the previously discussed techniques.

3 TECHNICAL APPROACH

As discussed in Chapter 2, control law design for high performance aircraft presents challenging and unique problems to the designer. Traditional techniques have proven to be either prohibitively costly in terms of the effort required in tuning and the complexity of developing a multi-region linearized gain scheduling design, or have simply sacrificed performance for ease of design. This chapter formally presents an innovative method of integrating an adaptive component with a learning component to form a new hybrid control law. The hybrid system is presented by introducing each component separately and then combining the components in a synergistic arrangement to form a superior flight control system.

3.1 ADAPTIVE CONTROL COMPONENT

Numerous adaptive control techniques have been developed for nonlinear systems with unmodeled dynamics or model uncertainty (Astrom & Wittenmark (1989), Slotine & Li (1991)). One major class of adaptive control, model reference adaptive control (MRAC), is considered in this thesis. The majority of MRACs can be grouped into two general categories, namely, direct and indirect adaptive control. Direct adaptive control approaches are characterized by the synthesis of control signals directly from observed plant behavior without the benefit of an explicit plant model. In contrast, the indirect adaptive control methods rely heavily on an explicit plant model. The control law for an indirect technique employs a local plant model that is updated from observed plant behavior. Although developing and periodically updating a plant model is not without its own costs, indirect techniques have the advantage that many different control designs

ATTACHMENT 2

techniques that are based on explicit plant models can be used. In either case, the adaptive control system reacts to differences between desired and predicted behavior by adjusting internal parameters to achieve desired closed-loop performance. These differences in behavior are typically attributed to nonlinearities, unmodeled dynamics, model uncertainty, or exogenous disturbances.

Although conventional adaptive control methods have the ability to stabilize and control some nonlinear systems, the closed-loop system is often unable to match the performance of a well-designed and well-tuned gain scheduled controller. This difference in performance stems from inherent time-delays or lags associated with adaptive controllers. Typically, the process of updating parameters of an adaptive control law requires several control intervals to accurately detect and compensate for variations in the plant behavior. Sensor noise exacerbates this dilemma since the required filtering creates additional lag. Adaptive control approaches also have performance limitations when presented with quasi-static state dependent disturbances. In particular, since adaptive controllers are reactive by nature, they are unable to learn and subsequently predict state dependent behavior (Baker & Farrell (1992)). Even if the plant repeatedly experiences the same disturbance at a particular location in the state space, the adaptive controller must wait until the effects of the discrepancy are observed before it can initiate changes in the parameters. Hence, adaptive controllers fail to make complete use of experientially gained knowledge. As will be discussed in a following section, this inadequacy of adaptive control can be overcome with the addition of a learning component.

The primary role of the adaptive control component in the hybrid system is to accommodate unmodeled dynamical behavior (i.e., behavior that is not expected based on the design model). Additionally, the adaptive component of the hybrid system has the auxiliary task of presenting estimates of any observed unmodeled state dependent dynamic behavior to the learning component (i.e., unknown dynamics that are a function of state in areas of the state space where the learned mapping can be improved).

ATTACHMENT 2

3.1.1 Control Law Derivation

Consistent with the above discussion, any adaptive control approach that is applicable to nonlinear dynamic systems with model uncertainty and that develops estimates of unknown state-dependent components of the plant dynamics is a candidate for the adaptive component in the hybrid control system. Adaptive techniques that require small amounts of on-line computation are especially appealing since extra computing power will be required to train the learning component. One such adaptive control technique is based on Time Delay Control (TDC). Developed by Youcef-Toumi (1990), TDC is an indirect adaptive technique designed for the class of systems with discrete nonlinear dynamics represented in the following form:

$$\mathbf{x}(k+1) = \mathbf{g}\{\mathbf{x}(k), k\} + \mathbf{h}\{\mathbf{x}(k), k\} + \mathbf{\Gamma}\mathbf{u}(k) + \mathbf{d}(k), \quad (3.1)$$

Notationally, \mathbf{g} and \mathbf{h} represent known (modeled) and unknown nonlinear plant dynamics vectors, respectively. These vectors are functions of the state \mathbf{x} and discrete time k . Furthermore, \mathbf{d} is a possibly time-varying disturbance vector. The control vector is represented by \mathbf{u} and is *linearly* through the control input matrix $\mathbf{\Gamma}$ of the new state. It will be assumed here (in this subsection only) that $\mathbf{\Gamma}$ is known without error. Section 3.2 addresses the issue of uncertainty in $\mathbf{\Gamma}$.

Overview of TDC

TDC uses a simple estimation algorithm to detect and compensate for unknown dynamics and time-varying disturbances. By examining the difference in the dynamical behavior between the actual state of the plant and that expected (given knowledge of the state and control at the previous time step and the modeled terms of Equation (3.1)), TDC constructs a combined estimate of the unknown dynamics \mathbf{h} and disturbances \mathbf{d} at the current time. Using this estimate (formed from state and control information at the previous time step), it is possible to form a control law that attempts to cancel the undesired

ATTACHMENT 2

known dynamics, the estimated unknown dynamics, and the estimated disturbances. Desired state dynamics can then simply be "inserted" along with a proportional error term to achieve desired tracking error dynamics.

Critical to the TDC control law is the method of obtaining the estimates of the unknown dynamics and unexpected disturbances. By employing information from the previous time step, TDC is able to react rapidly to changes in the dynamical behavior of the plant. This characteristic is ideal for systems that operate in an environment with large variations in the unknown dynamics and unexpected disturbances. However, this beneficial feature is not without some cost. Since TDC basically "differentiates" the state in arriving at the control action, any sensor noise affecting the observed values of the state and control will be amplified, resulting in noisy control signals and possible rate or position saturation of the actuators. This effect translates into poor performance and possibly to instabilities. To counter the effects of noise, filters are used. Although filters can accommodate noise, they add additional lag which reduces the performance of the adaptive system.

Development of TDC

The full development of the TDC control law is contained in Youcef-Toumi & Osamu (1990). For the sake of completeness, the fundamental equations are summarized below.

Assume that the plant can be written in the following form:

$$\mathbf{x}(k+1) = \Phi \mathbf{x}(k) + \mathbf{h}\{\mathbf{x}(k), k\} + \Gamma \mathbf{u}(k) + \mathbf{d}(k) \quad (3.2)$$

where \mathbf{x} is an n dimensional state vector, \mathbf{u} is an m dimensional vector of control inputs, Φ is an n by n state transition matrix, Γ is an n by m control weighting matrix, and \mathbf{h} and \mathbf{d} are n dimensional unknown state dynamics and disturbance vectors respectively. Notice that Equation (3.2) is a special case of Equation (3.1), since the current state acts linearly on the new state. Here, $\Phi \mathbf{x}(k)$ can be viewed as the best time invariant, linear

ATTACHMENT 2

approximation of the known function $g\{x(k), k\}$, linearized about a selected operating condition. This assumption essentially shifts plant nonlinearities and time dependencies to the unknown dynamics term h .

Define a desired n dimensional reference trajectory x_m to be the following linear, time-invariant system:

$$x_m(k+1) = \Phi_m x_m(k) + \Gamma_m r(k) \quad (3.3)$$

where Φ_m is an n by n reference state transition matrix, Γ_m is an n by m reference model command weighting matrix, and $r(k)$ is an m dimensional vector of reference commands. There is no requirement that the reference model be a linear, time-invariant system. Moreover, it is assumed that the reference command $r(k)$ is constrained in a way that the desired reference trajectory is achievable by the system described by Equation (3.2).

The difference between the desired reference state and plant state is the error vector:

$$e(k) = x_m(k) - x(k) \quad (3.4)$$

The control objective of TDC is to force this error vector to zero with the following desired error dynamics defined in terms of an error dynamics transition matrix Φ_e :

$$e(k+1) = \Phi_e e(k) \quad (3.5)$$

By expressing Φ_e in terms of Φ_m , the error dynamics can be written as

$$\Phi_e = \Phi_m + K \quad (3.6)$$

where K can be viewed as an error feedback matrix.

The control signal u that yields the desired error dynamics is obtained by incrementing Equation (3.4) one time step forward and substituting Equations (3.2) through (3.6) as follows

$$\begin{aligned} e(k+1) &= x_m(k+1) - x(k+1) \\ \Phi_e e(k) &= \Phi_m x_m(k) + \Gamma_m r(k) - \Phi x(k) - h\{x(k), k\} - \Gamma u(k) - d(k) \\ \Gamma u(k) &= \Phi_m x_m(k) + \Gamma_m r(k) - \Phi_m x(k) - h\{x(k), k\} - d(k) - \Phi_e e(k) \\ \Gamma u(k) &= [\Phi_m \quad \Phi] x(k) + \Gamma_m r(k) - h\{x(k), k\} - d(k) - K e(k) \end{aligned} \quad (3.7)$$

ATTACHMENT 2

Notice that the terms \mathbf{h} and \mathbf{d} on the right hand side of Equation (3.7) are unknown. They will be replaced by estimates $\hat{\mathbf{h}}$ and $\hat{\mathbf{d}}$. In particular, if \mathbf{h} and \mathbf{d} change relatively slowly, then their estimated value can be obtained by solving Equation (3.2) at the previous time step, yielding an estimate of the sum of the two terms \mathbf{h} and \mathbf{d} :

$$\begin{aligned}\hat{\mathbf{h}}\{\mathbf{x}(k), k\} + \hat{\mathbf{d}}(k) &= \mathbf{h}\{\mathbf{x}(k-1), k-1\} + \mathbf{d}(k-1) \\ &= \mathbf{x}(k) - \Phi\mathbf{x}(k-1) - \Gamma\mathbf{u}(k-1)\end{aligned}\quad (3.8)$$

Here we assume full knowledge of the state and control values $\mathbf{x}(k)$, $\mathbf{x}(k-1)$, and $\mathbf{u}(k-1)$.

Unless Γ^{-1} exists, which implies that $n = m$ so that the number of inputs equals the number of states, Equation (3.7) will not have a general, exact solution. Nevertheless, an approximate solution can be generated as follows

$$\mathbf{u}(k) = \Gamma^+ \left[(\Phi_m - \Phi)\mathbf{x}(k) + \Gamma_m \mathbf{r}(k) - \mathbf{h}\{\mathbf{x}(k), k\} - \mathbf{d}(k) - \mathbf{K}\mathbf{e}(k) \right] \quad (3.9)$$

where Γ^+ is the pseudo-inverse of Γ . The use of the pseudo-inverse of the control weighting matrix is necessitated by the fact that the majority of control systems have more states than controls. The following pseudo-inverse

$$\Gamma^+ = [\Gamma^T \Gamma]^{-1} \Gamma^T \quad (3.10)$$

results in the minimization of the L_2 norm $\|\Gamma\Gamma^+ - \mathbf{I}\|_2$.

Substituting Equations (3.8) into Equation (3.9) results in the TDC control law

$$\begin{aligned}\mathbf{u}(k) &= -\Gamma^+ \mathbf{K}\mathbf{e}(k) && \text{(error feedback)} \\ &+ \Gamma^+ [\Phi_m - \Phi]\mathbf{x}(k) && \text{(state feedback)} \\ &+ \Gamma^+ \Gamma_m \mathbf{r}(k) && \text{(command feedforward)} \\ &- \Gamma^+ [\hat{\mathbf{h}} + \hat{\mathbf{d}}] && \text{(cancellation)}\end{aligned}\quad (3.11)$$

The first term in Equation (3.11), *error feedback*, represents proportional feedback of the error between the desired and actual state at time k . The *state feedback* term determines the contribution of the state at discrete time k to the control. This term is a function of the difference between the desired trajectory dynamics and the linearized approximation of the plant dynamics. Commands enter the control law through the *command feedforward*

term. As compared to the feedback terms, the command term is feedforward in the sense that it is an open-loop term that is not a function of plant state. The *cancellation* term attempts to cancel the unknown dynamics and disturbances at the present time k by using approximations based on observed behavior at the previous time $k-1$.

3.1.2 Implementation Issues

The design parameters of the TDC control law include those associated with the reference model dynamics $\{\Phi_m, \Gamma_m\}$ and desired tracking error dynamics Φ_e (or equivalently the error feedback matrix $\mathbf{K} = \Phi_e - \Phi_m$). Of course, these parameters cannot be selected in an arbitrary manner. As alluded to in the previous section, TDC requires the use of a pseudo-inverse in the control law calculation due to the fact that the majority of systems have more states than controls. Hence, the control weighting matrix is singular and cannot be exactly inverted. By inserting Equation (3.11) into (3.2), the following constraint must be met in order for the plant state to track the model state with the desired error dynamics:

$$\{\mathbf{I} - \Gamma\Gamma^+\} \{[\Phi_m - \Phi]\mathbf{x}(k) + \Gamma_m \mathbf{r}(k) - \mathbf{h}\{\mathbf{x}(k), k\} - \mathbf{d}(k) - \mathbf{K}\mathbf{e}(k)\} = 0 \quad (3.12)$$

Notice that if Γ is square and invertible, then the first factor on the left hand side guarantees that the constraint is always met. If this is not the case, then values for the design parameters Φ_m , Γ_m , and \mathbf{K} must be selected to minimize the error of Equation (3.12) for arbitrary \mathbf{r} , \mathbf{h} , and \mathbf{d} . Alternatively, Γ^+ can be selected so that the nonzero second factor on the left hand side of Equation (3.12) is in the nullspace of $\{\mathbf{I} - \Gamma\Gamma^+\}$. However, the approaches for meeting the constraint of Equation (3.12) when Γ is non-square are generally difficult.

Beyond this constraint issue, the error feedback matrix \mathbf{K} is chosen to achieve the desired error dynamics Φ_e . Typically, error dynamics have been chosen as a function of the reference model dynamics (e.g., twice as fast). However, other selections can be

accommodated as long as the error dynamics are stable.

Selection of a desired reference model $\{\Phi_m, \Gamma_m\}$ is frequently application specific. Although there is no requirement on the method used to generate a reference model for a flight control application, typical design specifications are often stated in terms of characteristics of linear, time-invariant (LTI) systems. For example, military aircraft must meet MIL-F-8785C (1980) specifications for natural frequency and damping ratio of their characteristic modes. Thus, a LTI system is often employed in the role of a reference model. The reference model for the aircraft control problem addressed by this thesis is discussed in Section 4.3.3.

3.2 LEARNING CONTROL SYSTEM

The purpose of the learning system in the hybrid control law is to synthesize a mapping between the state and controls of the plant and an estimate of the unknown dynamics \hat{h} generated by an adaptive component. As discussed previously, connectionist networks have demonstrated an ability to learn highly nonlinear, multivariable mappings. In this section, the complete development of the learning system employed in the hybrid controller is presented.

3.2.1 Incremental Learning and Fixation

Since the objective of the network in control applications is to synthesize a mapping over a continuous input space, the training cost function in Equation (2.2) cannot be used directly (i.e., the number of training examples is not a finite set). As a result, one common approach for systems with a continuous input space is to use *incremental learning* (Baker & Farrell (1991), Rumelhart, *et al.* (1986)). Incremental learning algorithms seek to reduce a cost function defined in terms of the current input point rather than a cost function defined over a fixed set of samples as in Equation (2.2). Using this approach, the cost

ATTACHMENT 2

function defined in Equation (2.2) reduces to the single term

$$J = \frac{1}{2} [\mathbf{d}(\mathbf{x}) - \mathbf{f}_{ner}(\mathbf{x}, \mathbf{p})]^T [\mathbf{d}(\mathbf{x}) - \mathbf{f}_{ner}(\mathbf{x}, \mathbf{p})] \quad (3.13)$$

where $\mathbf{d}(\mathbf{x})$ is the desired network output at current state \mathbf{x} and $\mathbf{f}_{ner}(\mathbf{x}, \mathbf{p})$ is the output of the network as a function of state and parameters \mathbf{p} . The general parameter vector \mathbf{p} is used in Equation (3.13) to allow for nodal functions that are not simply a function of the state and weights. An incremental learning approach essentially provides a convenient, point-wise contribution to an aggregate cost function similar to Equation (2.2) since it can be computed quickly and efficiently.

During incremental learning, care must be taken to ensure that samples are sufficiently distributed throughout the input space so that over a finite period of time, the individual point-wise contributions of Equation (3.13) collectively provide an approximation to a batch-type cost function in Equation (2.2). Since parameters are updated at each sample, the network reacts to mapping errors at the current input. Unfortunately, sigmoidal networks possess a relatively high degree of "generalization," where parameter changes impact the network mapping over potentially large regions of the input space. As a consequence, the localized nature of incremental learning can result in "fixation" of the network, where the network attempts to achieve an accurate mapping at the current state, while potentially degrading an acceptable mapping already learned in other regions of the input space (Baker & Farrell (1992)).

The magnitude of the fixation problem is determined by the rate of mapping degradation in outlying regions *relative* to the time required to receive samples from all regions of input space. This rate of outlying mapping degradation is in turn determined by the degree of generalization and the learning rate. A network with a high level of generalization requires rapid and extensive distribution of sampling points or a very small learning rate to avoid problems associated with fixation. For control problems, the former is generally not possible since the sampling process is constrained by the system dynamics

ATTACHMENT 2

Furthermore, extensive investigation of the state space is typically inconsistent with the control objectives. This point is most evident in regulation, where the goal is to keep the system near some operating point. Due to such constraints on the sampling process, an alternative approach to avoiding fixation during incremental training is to reduce the degree of global generalization in the network. Such spatially localized networks are discussed in the next section.

3.2.2 Spatially Localized Learning

The basic idea of spatially localized learning is that experience (learning) in a local region of the input space should only affect the mapping in that particular locality, with a marginal effect in all other areas. Spatially localized learning prevents knowledge that has already been collected in other regions of the mapping from becoming incorrectly perturbed (i.e., corrupted). This is accomplished by lessening the extent of generalization to include only a local region. Figure 3.1 illustrates the concept of spatially localized learning. Let $f_{net}(x; p^0, p^1, \dots, p^N)$ represent the mapping to be learned, where x is the input vector and p^0, \dots, p^N are a set of parameters to be learned that define the mapping.

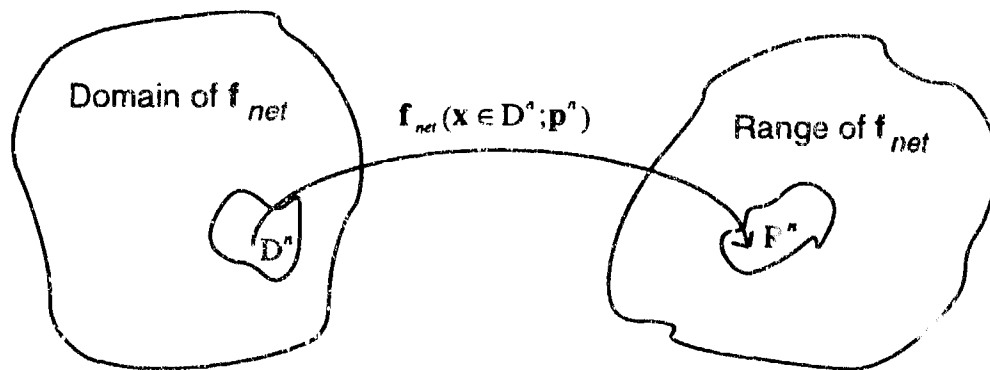


Figure 3.1 Spatially Localized Learning: the Ideal Case

Figure 3.1 shows a region of the domain D^n of the function to be learned being mapped to an associated region of the range R^n . The ideal situation for localized learning, as indicated in the figure, is that this local mapping be a function of a subset of the parameter set

denoted \mathbf{p}^n . Thus, the learning based on samples in D^n will only cause the subset of parameters \mathbf{p}^n to change. Of course, this represents the ideal situation which is not practical for a variety of reasons. However, the objective is that the result of learning from input samples in one region of the domain should not significantly alter previously learned mappings in distant regions of the domain.

This local generalization property of spatially localized learning contrasts with typical structures (e.g., sigmoidal networks) that are characterized by a much larger, more global generalization. The following discussion introduces and develops one example of a spatially localized learning system that is used in the hybrid control system. Learning is accomplished by an incremental gradient descent learning algorithm.

3.2.3 The Linear-Gaussian Network

One learning system design that exhibits spatially localized properties is the linear-Gaussian network. The linear-Gaussian network is an example of a local basis / influence function system (Baker & Farrell (1992), Millington (1991)). The network mapping is constructed from a set of hyperplanes that act as "basis" functions over a localized region of the input space. Although many functions could be used as a local basis, hyperplanes offer an attractive choice for the control problem due to their simple structure and similarity with conventional gain scheduled mappings. The influence function associated with each local basis function is an elliptic hyper-Gaussian. As the name suggests, the role of the influence function is to determine the region of applicability of a particular local basis function in the input space. For example, a basis function associated with a hyper-Gaussian whose center is very close to the current input plays a much larger role in the determination of the output of the mapping than a basis function whose Gaussian is centered far away from the current input. The following discussion details the linear-Gaussian network.

Node Descriptions

The local basis function of a linear-Gaussian node is formed by adding the weighted sum of the inputs with a bias. Equation (3.14) shows the relationship between the i^{th} linear basis function L_i and its inputs, \mathbf{x} :

$$\mathbf{L}_i(\mathbf{x}) = \mathbf{W}_i(\mathbf{x} - \mathbf{x}_{io}) + \mathbf{b}_i \quad (3.14)$$

Here, if n is the number of node inputs and m is the number of node outputs, then \mathbf{L}_i is an m dimensional vector, \mathbf{x} is a n dimensional vector of node inputs, \mathbf{W}_i is an $m \times n$ matrix whose elements are the weights on the input, \mathbf{x}_{io} is a n dimensional vector that represents the center of the Gaussian nodal function described below, and \mathbf{b}_i is a m dimensional bias vector.

The linear-Gaussian node uses a hyper-Gaussian as an influence function for the basis \mathbf{L}_i in Equation (3.14). The value for the i^{th} Gaussian function G_i is given by:

$$G_i(\mathbf{x}) = \exp\left[-\frac{1}{2}(\mathbf{x} - \mathbf{x}_{io})^T (\mathbf{D}_i)^2 (\mathbf{x} - \mathbf{x}_{io})\right] \quad (3.15)$$

where \mathbf{D}_i is a diagonal matrix containing values for the spatial decay of the Gaussian, \mathbf{x}_{oi} is the Gaussian center, and \mathbf{x} is the input vector. Figure 3.2 contains an illustration of a typical Gaussian function.

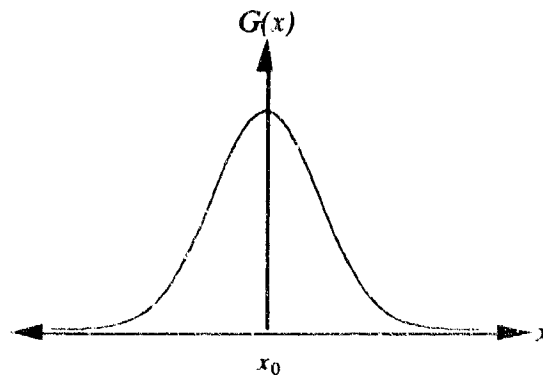


Figure 3.2 Gaussian Function

The Gaussian is a continuous function with finite asymptotic values. Moreover, a

ATTACHMENT 2

Gaussian is differentiable over the entire input space, which is important in any learning algorithm that relies on partial derivative information for training (e.g., gradient methods). The output of the linear-Gaussian node is simply the product of the linear basis function and the Gaussian influence function. The general structure is shown in Figure 3.3

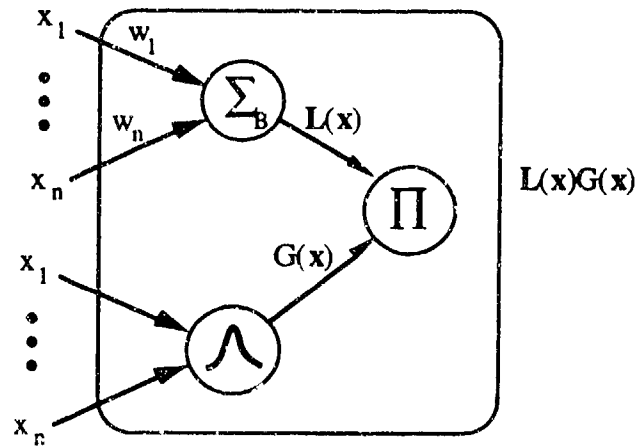


Figure 3.3 Linear Gaussian Node

where Σ_B represents a summing node with bias and Π a multiplication node. By dividing the i^{th} Gaussian function by the sum of all the Gaussians, the resulting quotient is the normalized i^{th} influence function, Γ_i . This relationship is shown in Equation 3.16 below

$$\Gamma_i(\mathbf{x}) = \frac{G_i(\mathbf{x})}{\sum_{i=1}^n G_i(\mathbf{x})} \quad (3.16)$$

where

$$\sum_{i=1}^n \Gamma_i(\mathbf{x}) = 1 \quad \text{and} \quad 0 < \Gamma_i(\mathbf{x}) \leq 1 \quad (3.17)$$

Combining Equations (3.14) through (3.16) yields the following equation for the m dimensional vector output of a linear-Gaussian network:

$$\mathbf{Y}(\mathbf{x}) = \sum_{i=1}^n \mathbf{L}_i(\mathbf{x}) \Gamma_i(\mathbf{x}) \quad (3.18)$$

ATTACHMENT 2

Network Architecture

Linear Gaussian networks use a feedforward architecture and consist of three main layers: input, hidden, and output as depicted in Figure 3.4. The first layer of the network, the input layer, simply passes the input values to the subsequent hidden layer. As one would expect, there are the same number of input nodes as there are network inputs. The hidden layer is not directly observable to the external environment. This hidden layer contains two elements, namely the linear-Gaussian nodes and nodes that normalize the Gaussian influence function. By adding enough linear-Gaussian nodes, a single hidden layer network can provide arbitrarily accurate function approximations. Furthermore, multiple hidden layers of linear-Gaussian nodes lead to non-localized mappings. For these reasons, only linear-Gaussian networks with a single hidden layer are investigated. The final layer is the output layer. It contains as many nodes as there are outputs. A typical linear-Gaussian network is shown in Figure 3.4

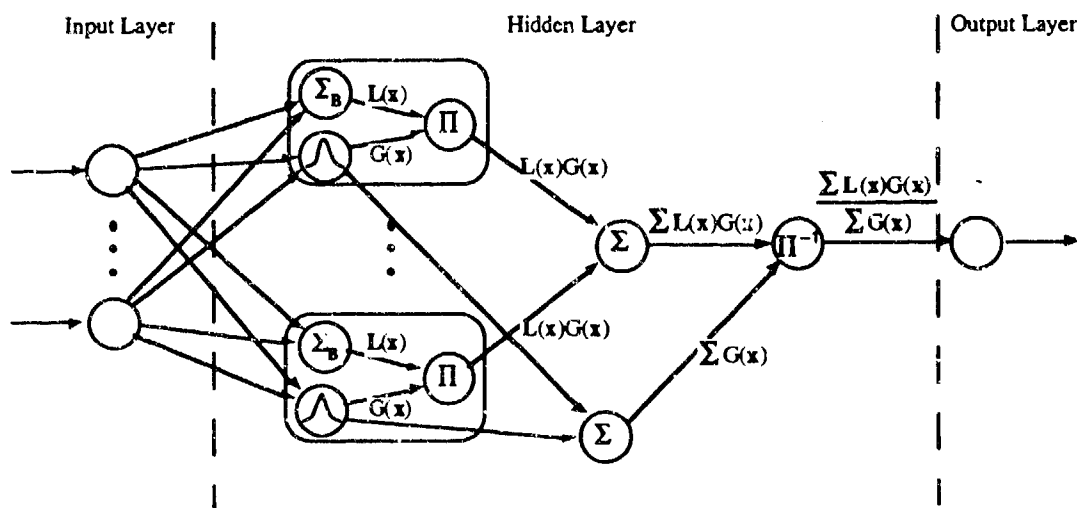


Figure 3.4 Multi-Input, Single Output Linear Gaussian Network

where the negative sign of the rightmost Π node indicates that the argument is reciprocated prior to the multiplication.

Learning Algorithm

The linear-Gaussian network uses a supervised, incremental gradient descent algorithm to adjust the network parameters in the negative direction of their gradient with the cost function:

$$\mathbf{p}_i(k+1) = \mathbf{p}_i(k) - \alpha \left. \frac{\partial J}{\partial \mathbf{p}_i} \right|_k ; \quad \alpha > 0 \quad (3.19)$$

where \mathbf{p}_i is a vector of the adjustable parameters of the i^{th} node (e.g., weight matrix elements, bias, spatial decay, or center) and J is the cost at a particular training sample, and α is the learning rate. The typical cost used for linear-Gaussian networks is shown in Equation (3.20)

$$J = \frac{1}{2} [\mathbf{d}(\mathbf{x}) - \mathbf{f}_{\text{net}}(\mathbf{x}, \mathbf{p})]^T [\mathbf{d}(\mathbf{x}) - \mathbf{f}_{\text{net}}(\mathbf{x}, \mathbf{p})] \quad (3.20)$$

where \mathbf{d} is the desired output as a function of input state \mathbf{x} , and \mathbf{f} is the output of the network as a function of input state and network parameters \mathbf{p} . In minimizing the cost at each step (i.e., for each training sample), all of the parameters, or just a subset, can be adjusted using Equation (3.19). The local learning rate for each parameter can be adjusted independently in order to achieve a more rapid convergence.

Besides the basic nodal and architectural differences, the learning algorithm of the linear-Gaussian network also differs from that of the classic sigmoidal network. Since the normalized Gaussian influence functions represent a measure of the significance that each node has on a particular value of the input \mathbf{x} (i.e., the influence of each node on the output for a given input), it is reasonable to eliminate insignificant nodes from the learning calculation. Due to the elimination of the nodes, the computational efficiency and hence the training time of the network are improved. For example, for a given input value \mathbf{x} , the learning algorithm might first order the Gaussian nodes by their normalized influence and use only enough nodes so that the sum of the normalized influences equals or exceeds some threshold value (e.g., 99%). Since the remaining nodes have only a minor effect on

ATTACHMENT 2

the local region, their outputs need not be included in the parameter update. For large networks, this can result in a substantial reduction in computation.

Application Issues

The number of nodes needed by a linear-Gaussian network is dependent on the characteristics of the function it is attempting to approximate and on any requirements placed on the desired rate of convergence and the level of acceptable errors in the learned mapping. Although no set of strict rules has been developed for selecting the number of nodes, several guidelines do exist. For functions that are very smooth, the mapping can be realized with relatively few nodes spread evenly throughout the input space. A large number of nodes will not improve this mapping and will only serve to increase the network training time. However, more complex functions with large local variations will require a large number of nodes, each with a relatively small sphere of influence.

The sphere of influence of a Gaussian function is determined by the spatial decay matrix. Hence, the spatial decay matrix is a factor in determining the size of the local regions in the input space. If the spatial decay is large, the transitions between the regional linear basis functions will be more abrupt if the density of basis functions is not high. This property is ideal for more complex functions. However, a large spatial decay will require many more nodes to sufficiently map the entire input space. In contrast, small spatial decay rates result in large regions of influence that are ideal for smooth, slowly varying functions.

Initial values for the weights, biases, and Gaussian centers must also be selected. The basis function described by a weight matrix and bias vector represent a best guess of the desired mapping based on *a priori* information. Hence, values for the weights and biases can be initialized from an existing gain scheduled controller or other linearizable control law. In cases with considerable *a priori* knowledge, the adjustable parameters are presumably much closer to their optimal values, and training time is greatly reduced. If no *a priori* knowledge is available about the mapping, the weights and biases are set to zero.

The initialization of the Gaussian centers effectively locates the influence functions in the input space, and generally, the centers are placed so that the entire input space is spanned.

In summary, a linear-Gaussian network is one example of a spatially localized learning system. This network combines linear basis functions with Gaussian functions to provide the properties of local learning and the generalization properties of typical connectionist networks. Networks that employ spatially localized learning are required for control systems that regularly encounter scenarios that might cause fixation, as described in Subsection 3.2.1. Although linear-Gaussian networks tend to require more nodes and thus more memory (due to localization), improved learning efficiency and, more importantly, better functional mappings can be obtained.

3.3 HYBRID LEARNING / ADAPTIVE CONTROL

The hybrid control law developed in this section represents one approach to combining a learning system with an adaptive component with the objective of improving performance in the presence of unmodeled dynamics and model uncertainty. In augmenting an indirect adaptive controller with a connectionist learning system, the general goal is to develop a control law that combines the strengths of each component.

3.3.1 Hybrid Control System Architecture

Adaptive control systems are capable of controlling complex dynamic systems. However, traditional adaptive control techniques only react (after the fact) to differences between actual and expected behavior — they have no anticipatory capacity. Learning in connectionist systems is fundamentally a process of function approximation. Hence, given the vehicle state and the applied control as an input and the unknown dynamics as desired outputs, a connectionist learning system is capable of realizing a mapping of the state and control dependent dynamics. Thus, by augmenting an adaptive controller with a learning

ATTACHMENT 2

system, it is possible to anticipate the state dependent components of the plant dynamics by "looking up" the values of that component of the dynamics for the current situation from the network, instead of waiting for an adaptive component to react to observed differences between the actual and expected state values. *By incorporating a learning system into the control law, the hybrid controller is able to use experientially gained knowledge.* Figure 3.5 illustrates the control system architecture of the hybrid adaptive / learning controller (Baker & Farrell (1991)).

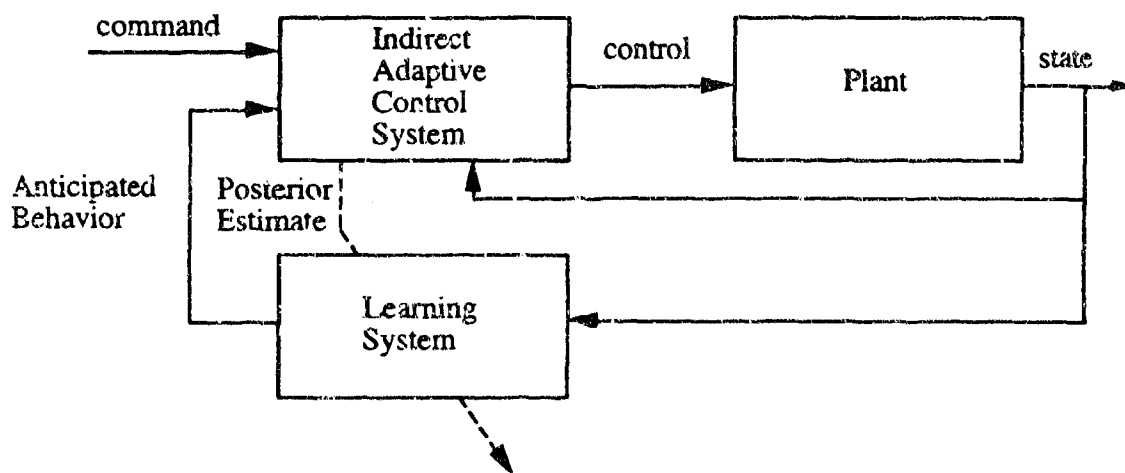


Figure 3.5 Hybrid System Architecture

The role of each component in the hybrid system is straightforward. The adaptive component provides an adaptive capability to accommodate unmodeled dynamic behavior that is not expected (based on the design model). Moreover, it provides a posterior estimate of any unmodeled state and control dependent behavior which can be used to train the learning system. The role of the learning system is to anticipate vehicle behavior that varies predictably with the current state and control.

3.3.2 Learning Versus Adaptation

Since both the adaptive component and the learning component in the hybrid control system are based on parameter adjustment algorithms that use information gained by

ATTACHMENT 2

observing the closed-loop behavior of the plant, one might think it is difficult to distinguish between the two components. However, for the majority of systems, distinguishing qualities do exist. The following discussion presents the different goals and characteristics of the adaptive and learning components in an attempt to differentiate the two.

The adaptive component of the hybrid system can be characterized by its reactive approach to accommodating local disturbances and *apparent* time-varying dynamics. Nonlinearities that are a function of the operating condition of the plant appear to the adaptive component as time-varying dynamics when they are actually changes in the local linearized behavior of a nonlinear, time-invariant plant. Since adaptive controllers typically lack the ability to associate the required changes in the control action as a function of the operating conditions, the controller must continually adapt to all unexpected effects, even those which are experienced repeatedly and are actually due to time-invariant nonlinearities. In other words, adaptive controllers have no "memory" and are unable to anticipate dynamics that are strictly a function of state. Thus, this lack of memory prevents any anticipatory action by the controller. Moreover, to prevent a situation where the adaptive controller is continuously in some suboptimal, partially adapted state, the generation of the unknown dynamics estimate must be relatively fast when compared to the plant dynamics. In summary, the adaptive component reacts to unexpected effects in order to maintain locally desired behavior; it is best at accommodating novel situations and slowly time-varying dynamics.

The reactive characteristics of the adaptive component directly contrasts with the constructional emphasis of the learning component. In particular, the objective of the learning component in the hybrid control law is to associate initially unknown state dependent dynamics with the state and control at the current operating condition. The association is essentially a memory function (or mapping) that stores experientially gained knowledge. This knowledge of originally unknown dynamics can be exploited by the hybrid control system as a means of anticipating transient behavior instead of waiting to

ATTACHMENT 2

react to errors observed in the output. Moreover, this allows the adaptive component to focus on accommodating slowly varying exogenous (not state dependent) disturbances. Since the objective of learning is to realize a mapping of state dependencies over the *entire* operating envelope, the learning system is characterized by a global optimization and relatively slow dynamics. Table 3.1 summarizes the major differences between adaptation and learning (Baker & Farrell (1991)).

Table 3.1 Adaptation vs. Learning

| ADAPTATION | LEARNING |
|---|--|
| <i>reactive</i> : maintain desired behavior (local optimization) | <i>constructional</i> : synthesize desired behavior (global optimization) |
| temporal emphasis | spatial emphasis |
| no "memory" \Rightarrow no anticipation | "memory" \Rightarrow anticipation |
| fast dynamics | slow dynamics |

The goal of the hybrid controller is to combine the different behavioral characteristics of the adaptive and learning components in a synergistic fashion. Ideally, the adaptive controller accommodates local unmodeled dynamics and novel state dependencies, while the learning system is responsible for reducing state and control dependent model uncertainty.

3.3.3 Control Law Development

As discussed previously, TDC is one example of a particularly simple indirect adaptive control approach. Recall that TDC calculates an estimate of the sum of the unknown dynamics \mathbf{h} and disturbances \mathbf{d} at the previous time step by examining the difference in the dynamical behavior between the current state of the plant and the expected state given the state and control at the previous time step. Assuming that \mathbf{h} and \mathbf{d} do not change significantly over a control time step, TDC uses this old value of the sum of \mathbf{h} and

ATTACHMENT 2

\mathbf{d} in formulating the control law. By integrating TDC with a learning component to form a hybrid controller, this delay in the estimated value of \mathbf{h} can be eliminated. Although this delay is possibly insignificant with short control cycle times in the absence of sensor noise, such is not the case in a more realistic environment. If the control law is generated at a fast rate, the unknown dynamics and disturbances at the previous time will accurately reflect the current values (in the absence of noise). However, as the cycle time is increased, the potential for error in the estimates grows. If sensor noise is present, it is still possible to predict the current state within a given tolerance. However, since \mathbf{h} and \mathbf{d} are essentially found by taking the derivative of the state, sensor noise can have a large impact on these estimates and subsequently the control generated by TDC.

The most common technique for offsetting the effects of sensor noise is to use a filter. Note, however, that filtering the noise only adds to the delay already associated with \mathbf{h} and \mathbf{d} . For this reason, a hybrid approach can offer significant advantages due to the use of a connectionist learning system. Since sensor noise can alter the estimates of \mathbf{h} and \mathbf{d} significantly, it is possible to have conflicting desired output values for the same input (over time). Given this contrasting information, connectionist systems tend to learn the average value. Thus, if the sensor noise is zero mean, which is the assumed case, the correct mapping will still be realized by the learning system. Since the recall of the learned estimates of the state dependent dynamics is nearly instantaneous, *the hybrid system is essentially able to remove the delay associated with the adaptive component.*

As alluded to earlier, the hybrid control law can be derived by augmenting the TDC equations with a learning component. Assume the nonlinear plant can be written in the following form:

$$\mathbf{x}(k+1) = \Phi \mathbf{x}(k) + \Gamma \mathbf{u}(k) + \mathbf{f}_{net}\{\mathbf{x}(k), \mathbf{u}(k)\} + \mathbf{h}\{\mathbf{x}(k), \mathbf{u}(k), k\} + \mathbf{d}(k) \quad (3.21)$$

where $\mathbf{x}(k)$ is an n dimensional state vector at discrete time k , $\mathbf{u}(k)$ is an m dimensional control vector at k , Φ is an n by n state transition matrix, Γ is an n by m control weighting matrix, \mathbf{h} and \mathbf{d} are n dimensional unknown dynamics and disturbances vectors

ATTACHMENT 2

respectively, and f_{net} is the n dimensional learned component of the state dependent dynamics. Equation (3.21) differs from the form of the plant model used in the TDC derivation only by the learned dynamics term f_{net} . Moreover, the unknown dynamics term is allowed to be a function of control as well as state, essentially accounting for errors in the assumed (but unlikely to be) perfectly known Γ .

As before, the desired reference trajectory is given by

$$x_m(k+1) = \Phi_m x_m(k) + \Gamma_m r(k) \quad (3.22)$$

Here $x_m(k)$ represents the n dimensional desired model state vector at time k , Φ_m is the n by n state transition matrix defined by the linear relationship between the current and next state, $r(k)$ is the m dimensional command vector and Γ is the n by m model command weighting matrix. As was the case with the derivation of the TDC control law in Section 3.1.1, there is no requirement that the reference model be linear. The only requirement on the reference model is that the desired trajectory is achievable, otherwise the control law may saturate the effectors and yield unsatisfactory performance.

If the difference between the desired reference state and plant state at discrete time k is represented by the error vector

$$e(k) = x_m(k) - x(k) \quad (3.23)$$

then the control objective of the hybrid control law is to force this error vector to zero with the following dynamics

$$e(k+1) = [\Phi_m + K]e(k) = \Phi_e e(k) \quad (3.24)$$

where K is interpreted as the error feedback matrix and Φ_e is the desired error dynamics matrix.

If Equations (3.21) through (3.23) are substituted into Equation (3.24), the control signal u that yields the desired error dynamics is obtained from:

$$\Gamma u(k) = [\Phi_m - \Phi]x(k) + \Gamma_m r(k) - f_{net}\{x(k), u(k)\} - h\{x(k), k\} - d(k) - Ke(k) \quad (3.25)$$

To isolate the functions on the right hand side of Equation (3.25) that are dependent on \mathbf{u} at the current time k , approximations for the unknown dynamics and disturbances as well as the output of the connectionist network are made.

If \mathbf{h} and \mathbf{d} change relatively slowly, then their estimated value can be obtained (as before) by solving Equation (3.21) at the previous time step, yielding

$$\begin{aligned}\hat{\mathbf{h}}\{\mathbf{x}(k), \mathbf{u}(k), k\} + \hat{\mathbf{d}}(k) &= \mathbf{h}\{\mathbf{x}(k-1), \mathbf{u}(k-1), k-1\} + \mathbf{d}(k-1) \\ &= \mathbf{x}(k) - \Phi \mathbf{x}(k-1) - \Gamma \mathbf{u}(k-1) - \mathbf{f}_{net}\{\mathbf{x}(k-1), \mathbf{u}(k-1)\}\end{aligned}\quad (3.26)$$

The network function \mathbf{f}_{net} in Equation (3.25) can be approximated using the first-order Taylor series expansion shown in Equation (3.27) to isolate the linear dependence on \mathbf{u} at time k . Since the network is continuously differentiable, the Jacobian in Equation (3.27) is known to exist. Moreover, this Jacobian information is already calculated since it is needed for the learning algorithm discussed in Section 3.2.

$$\mathbf{f}_{net}\{\mathbf{x}(k), \mathbf{u}(k)\} \approx \mathbf{f}_{net}\{\mathbf{x}(k), \mathbf{u}(k-1)\} + \left. \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right|_{\mathbf{x}(k), \mathbf{u}(k-1)} \bullet (\mathbf{u}(k) - \mathbf{u}(k-1)) \quad (3.27)$$

Substituting these approximations into Equation (3.25) and solving for \mathbf{u} at the current time k (using a pseudo-inverse) yields the following hybrid control law:

$$\begin{aligned}\mathbf{u}(k) &= - \left[\Gamma + \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right]^+ \mathbf{K} \mathbf{e}(k) \\ &\quad + \left[\Gamma + \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right]^+ [\Phi_m - \Phi] \mathbf{x}(k) \\ &\quad + \left[\Gamma + \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right]^+ \Gamma_m \mathbf{r}(k) \\ &\quad - \left[\Gamma + \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right]^+ [\hat{\mathbf{h}} + \hat{\mathbf{d}}] \\ &\quad - \left[\Gamma + \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right]^+ \left[\mathbf{f}_{net}\{\mathbf{x}(k), \mathbf{u}(k-1)\} - \left. \frac{\partial \mathbf{f}_{net}}{\partial \mathbf{u}} \right|_{\mathbf{x}(k), \mathbf{u}(k-1)} \bullet \mathbf{u}(k-1) \right]\end{aligned}\quad (3.28)$$

The differences between the TDC control law in Equation (3.11) and the hybrid control law are the result of the added learning terms. The fifth term in Equation (3.28) represents the

ATTACHMENT 2

learned state dependent dynamics. The partial derivative of the network output with respect to the control used in the pseudo-inverse calculation is a linear correction for errors in Γ as discussed below.

Beyond removing the delay associated with a purely adaptive controller, the hybrid control system is able to reduce model uncertainty. This is accomplished by using partial derivative information for the learned network term with respect to the control inputs. For example, if there are errors in the coefficients of the assumed linear control weighting matrix Γ , or the control actually affects the next state in a nonlinear fashion, the partial of the learned dynamics with respect to the control represents the locally linearized unmodeled effect of the controls. Assuming accurate derivative information can be obtained from the network, the actual manner that the controls impact the next state is thus the assumed linear control weighting matrix corrected by this learned effect. *The technique of using the partial information to improve the a priori design and ultimately reduce model uncertainty represents a potentially major improvement over the TDC control law.*

4 EXPERIMENTS

A learning enhanced hybrid flight control system is demonstrated using the realistic model of a high-performance, supersonic aircraft that is described in Section 4.2. However, because the complexity of this aircraft model makes control system analysis difficult, the hybrid controller is first applied to a relatively simple nonlinear aeroelastic oscillator, described in Section 4.1. For this simple example, an exact truth model of the nonlinear plant dynamics is known, and the mapping that is synthesized by the control system can be compared to the known dynamics.

The objective of the experiments detailed in this section is to illustrate some of the properties of the hybrid control system. In particular, the goal is to demonstrate the ability of the hybrid system to improve the control of a nonlinear plant with model uncertainty and unmodeled dynamics that are a function of state and control. Both the aeroelastic oscillator and the high performance aircraft fall into this category. A secondary objective is to illustrate the learning characteristics of spatially localized connectionist networks when applied to control systems.

Section 4.1 and Section 4.2 each begins with a description of the plant model of interest (i.e., the aeroelastic oscillator and the high performance aircraft) and the physical motion that the model represents. This description is followed by a brief discussion of the open-loop dynamics and other characteristics of that model. Next, the reference model, along with the motivation for its selection, is presented. Issues in applying the hybrid control law to each plant are also discussed. This development is followed by two experiments for each plant that highlight the capabilities of the hybrid controller.

4.1 AEROELASTIC OSCILLATOR

4.1.1 Description

The aeroelastic oscillator models the motion of a square prism in a steady wind with an external control force. If the aeroelastic oscillator is constrained to translational motion normal to the incident wind, the dynamics resemble a classic mass-spring-dashpot system with an additional aerodynamic lift force due to an effective angle-of-attack between the wind and the prism (Parkinson & Smith (1963)). Figure 4.1 illustrates the aeroelastic oscillator model, where $V(t)$ is the incident wind, $L(t)$ is the aerodynamic lift force, $f(t)$ is the control force, m is the mass of the square prism, r is the damping coefficient, and k is the spring constant. The two state variables, position $x(t)$ and velocity $v(t)$, represent motion normal to the incident wind.

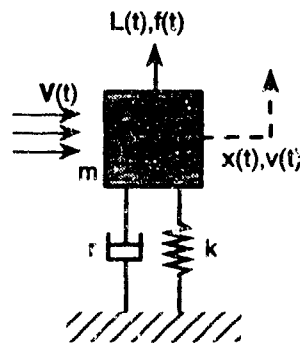


Figure 4.1 Aeroelastic Oscillator Model

The aerodynamic lift force is a nonlinear function of the effective angle-of-attack of the prism with respect to the incident wind. The effective angle-of-attack is due to the motion of the prism as illustrated in Figure 4.2, where α denotes the effective angle-of-attack and V_{REL} is the relative velocity.

ATTACHMENT 2

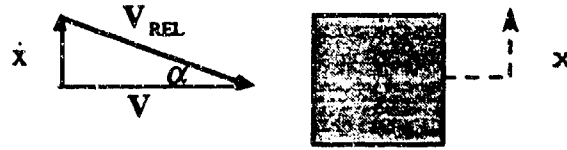


Figure 4.2 Effective Angle-of-Attack

Although current aerodynamic theory does not offer an analytic solution for the flow around a square prism, experimental data has been used to develop an approximation to the coefficient of lift (C_L) as a seventh-order polynomial of the effective angle-of-attack (Parkinson & Smith (1963)). Expressions for the coefficient of lift as well as for the resulting lift force L are given by:

$$C_L = A_1 \left(\frac{\dot{x}}{V} \right) - A_3 \left(\frac{\dot{x}}{V} \right)^3 + A_5 \left(\frac{\dot{x}}{V} \right)^5 - A_7 \left(\frac{\dot{x}}{V} \right)^7 \quad (4.1)$$

$$L = \frac{1}{2} \rho V^2 h l C_L \quad (4.2)$$

where the small angle approximation $\alpha = \dot{x} / V$ has been used, ρ is the air density, h is the side length of the prism, and l is the axial length of the prism. The differential equation governing the dynamics of the aeroelastic oscillator is:

$$m \frac{d^2 x}{dt^2} + r \frac{dx}{dt} + kx = L + f \quad (4.3)$$

Equation (4.3) can be nondimensionalized by dividing through by kh and making the following change of variables:

$$X = \frac{x}{h}; \quad n = \frac{\rho h^2 l}{2m}; \quad \omega = \sqrt{\frac{k}{m}}; \quad U = \frac{V}{\omega h}; \quad b = \frac{r}{2m\omega}; \quad \tau = \omega t$$

Applying this change of variables and substituting for the lift from Equation (4.2) yields:

$$\frac{d^2 X}{d\tau^2} + 2b \frac{dX}{d\tau} + X = nA_1 U \frac{dX}{d\tau} - \frac{nA_3}{U} \left(\frac{dX}{d\tau} \right)^3 + \frac{nA_5}{U^3} \left(\frac{dX}{d\tau} \right)^5 - \frac{nA_7}{U^7} \left(\frac{dX}{d\tau} \right)^7 + f \quad (4.4)$$

Equation (4.4) can be rewritten in a state space realization as:

ATTACHMENT 2

$$x_1 = X; \quad x_2 = \frac{dX}{d\tau} \quad (4.5)$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & (nA_1U - 2b) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} f + \begin{bmatrix} 0 \\ -\frac{nA_3}{U}x_2^3 + \frac{nA_5}{U^3}x_2^5 - \frac{nA_7}{U^7}x_2^7 \end{bmatrix} \quad (4.6)$$

where x_1 and x_2 are nondimensionalized position and velocity states, respectively.

Although the aeroelastic oscillator is a relatively simple second-order plant with a single control variable (force f), it still presents difficulties to conventional control design techniques due to the nonlinearities and uncertain parametric values (e.g., A_1, A_3, A_5, A_7) for the lift force. For these reasons, the aeroelastic oscillator has been selected to illustrate the properties of the hybrid controller.

4.1.2 Open Loop Dynamics

The nonlinearities in the open-loop dynamics of the aeroelastic oscillator in Equation (4.6) are a function of both mass velocity and incident wind velocity. For low incident wind velocities, the focus of the state trajectories in the phase plane is stable and the plant returns to the origin after exogenous disturbances. However, for higher wind velocities, the system tends to oscillate in a stable limit cycle. If the wind velocity is further increased, state trajectories in the phase plane are characterized by two stable limit cycles separated by an unstable limit cycle. Since the aeroelastic oscillator either returns to the origin or exhibits a stable limit cycle in face of disturbances for any value of incident wind, it is globally open-loop stable and a feedback loop is not required to provide nominal (bounded input / bounded output) stability.

4.1.3 Reference Model

As discussed in Section 3.3, the hybrid control law is designed to cause the plant state trajectory to follow a reference trajectory generated by a reference model. This reference model has a significant influence on the performance of the closed-loop system,

since by definition it represents the desired trajectory of the controlled vehicle states. As a result, if an unsatisfactory reference model is selected, the vehicle acting under the hybrid control law will also be unsatisfactory. Furthermore, if the reference model demands unrealistic state trajectories (e.g., reference trajectories that are chosen without regard to the limitations of the actual plant dynamics), control saturation leading to inadequate performance or even instabilities (in the general case) can occur. For these reasons, the reference model must be selected to yield satisfactory dynamics within the limitations of the vehicle or plant as required by specifications.

For the aeroelastic oscillator, the reference model was chosen to be the linear closed-loop system that results from applying an optimal linear quadratic control design to the aeroelastic oscillator dynamics linearized about the origin. The quadratic cost functional weights states and control equally. Thus, the objective of the hybrid control law is to force the true nonlinear model to behave identically to the linear reference model. Although not a requirement, a linear reference model is often used to achieve specifications (objectives) that have been stated in terms of natural frequency and damping ratio requirements. The reference model for the aeroelastic oscillator has been designed with a natural frequency of 1.12 radians per second and a damping ratio of 0.76.

4.1.4 Application of Hybrid Controller

To aid in the design, simulation, and analysis of the hybrid learning system, a custom-built software package developed at Draper Laboratory and coined "NetSim" was used. NetSim is a general-purpose simulation and design package that enables a variety of connectionist learning control systems to be developed interactively (Alexander, *et al.* (1991)). Through a graphical interface, pre-compiled code modules are connected in a block diagrammatic format to form the desired system. For dynamic systems, typical modules include plants, transforms (e.g., signal modifiers such as delays or switches), summing and gain objects and even dynamic compensators. NetSim also contains design

ATTACHMENT 2

tools that allow the user to create connectionist networks by graphically specifying the network nodes and architecture. All of the code modules are automatically linked together at run time, resulting in a complete system in which the outputs can be viewed on-line while the simulation is in progress.

The closed-loop simulation for the aeroelastic oscillator system uses four main modules as illustrated in the block diagram in Figure 4.3. This figure is a screen dump of the actual simulation window. The main modules include the reference model, hybrid controller, aeroelastic oscillator, and linear-Gaussian network. In addition to these main components, supporting operators are needed to modify the signals passed between the main modules to deliver the expected variables in the proper time sequence. The arrows between modules represent exchanges of variables, and the number in the lower left corner of each block dictates the order of execution at each time step. Modules called more than once per time step are shown with multiple sequence numbers.

Each module in Figure 4.3 performs a specific function in modeling the closed-loop dynamic system. The first module in the sequence is *Random*. *Random* outputs a randomly generated commanded position at the current time k . This command is held constant for a user specified amount of time. Once that time has elapsed, a new command is issued. *AO Reference* outputs the desired (model) state trajectory of the aeroelastic oscillator for the given command. The reference trajectory is generated using a discrete version of the optimal linear design as discussed above.

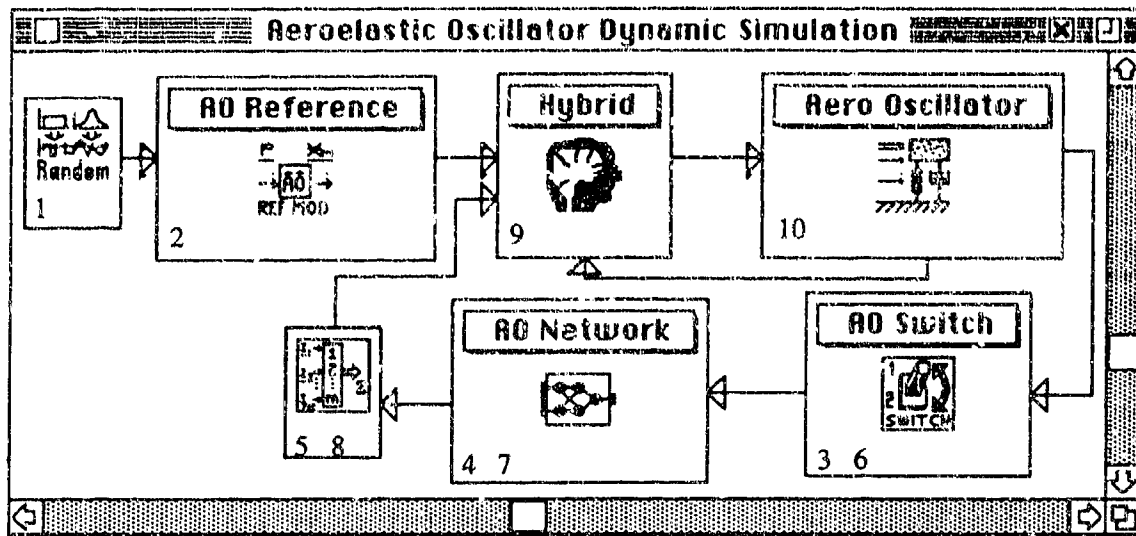


Figure 4.3 Block Diagram of the Aeroelastic Oscillator System

The *AO Switch* module supplies the network with the state and control at the appropriate time. It also sends a flag to the network to insure that learning only occurs with states and control at consistent times (e.g., learning occurs when the state, control, and desired output are all at the same time instant). *AO Network* is a linear-Gaussian network that serves as the learning system in the hybrid controller. The *Multiplexor* (shown with sequence numbers 5 and 8) gathers the outputs from the network that are needed for implementing the hybrid control law. *Hybrid* calculates the control signal based on the hybrid control law developed in Section 3.3. This control signal is passed to the *Aero Oscillator* module. The *Aero Oscillator* module contains the continuous nonlinear equations-of-motion of the plant. These equations are integrated using either an Euler or 4th order Runge-Kutte technique. The type and rate of integration, as well as plant parameters and initial conditions, are selected by the user. Table 4.1 summarizes the output of each module for one time step.

ATTACHMENT 2

Table 4.1 Module Execution Summary

| Sequence # | Module | Module Output |
|------------|--------------------|--|
| 1 | Random | $r(k)$ |
| 2 | AO Reference | $x_m(k+1) = \Phi_m x_m(k) + \Gamma_m r(k)$ |
| 3 | Switch | $x(k), u(k-1)$; Don't Learn Flag |
| 4 | AO Network | $f_{net}(x(k), u(k-1))$; $\frac{\partial f_{net}}{\partial u} \Big _{x(k), u(k-1)}$ |
| 5 | Multiplexor | $f_{net}(x(k), u(k-1))$; $\frac{\partial f_{net}}{\partial u} \Big _{x(k), u(k-1)}$ |
| 6 | Switch | $x(k-1), u(k-1)$; Learn Flag |
| 7 | AO Network | $f_{net}(x(k-1), u(k-1))$ |
| 8 | Multiplexor | $f_{net}(x(k), u(k-1))$, $\frac{\partial f_{net}}{\partial u} \Big _{x(k), u(k-1)}$, $f_{net}(x(k-1), u(k-1))$ |
| 9 | Hybrid | $u(k) = - \left[\Gamma + \frac{\partial f_{net}}{\partial u} \right]^+ Ke(k)$ $+ \left[\Gamma + \frac{\partial f_{net}}{\partial u} \right]^+ [\Phi_m - \Phi] x(k)$ $+ \left[\Gamma + \frac{\partial f_{net}}{\partial u} \right]^+ \Gamma_m r(k)$ $- \left[\Gamma + \frac{\partial f_{net}}{\partial u} \right]^+ [\hat{h} + \hat{d}]$ $- \left[\Gamma + \frac{\partial f_{net}}{\partial u} \right]^+ \left[f_{net}\{x(k), u(k-1)\} - \frac{\partial f_{net}}{\partial u} \Big _{x(k), u(k-1)} u(k-1) \right]$ |
| 10 | Aero Oscillator | $\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & (nA_1 U - 2b) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$ $+ \begin{bmatrix} 0 \\ -\frac{nA_1}{U} x_2^3 + \frac{nA_5}{U^3} x_2^5 - \frac{nA_7}{U^7} x_2^7 \end{bmatrix}$ |

4.1.5 Aeroelastic Oscillator Experiment 1

In experiment 1, a selected reference trajectory was repeated continuously in order to learn the state dependent, previously unknown dynamics f_{net} . The control objective was simply the regulation of both states about the origin given an initial position of -1 and velocity of 0.5 . By using the geometry and velocity parameters for a particular incident wind velocity found in Parkinson & Smith (1963), the equations-of-motion used in experiment 1 become:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 1.2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ -26.1x_2^3 + 127.3x_2^5 - 158.9x_2^7 \end{bmatrix} \quad (4.7)$$

The nonlinear terms Equation (4.7) were not supplied to the control system and represent the unknown dynamics in Equation (3.21).

Figures 4.4 and 4.5 illustrate the reference trajectories for position and velocity (based on the linear model described above) for the selected initial conditions and command. These reference trajectories represent the desired states at each time step, and any deviation from the reference by the actual states can be considered an error. The position and velocity trajectories of the nonlinear aeroelastic oscillator controlled by TDC alone (TDC Position / Velocity) are also shown in Figures 4.4 and 4.5 and are almost indistinguishable from the reference. In this case, the TDC controlled trajectories were produced by integrating the aeroelastic oscillator equations-of-motion at 200 Hertz and generating a control signal at that same rate. Moreover, there was no noise in the observed state and control values used by TDC. Combining these facts, it is not surprising that the TDC controller does extremely well in generating a control law that drives the plant along the reference trajectory. Indeed, because of the extremely small time step, the unknown dynamics observed at the previous time provide an accurate estimate of the unknown dynamics at the current time that is required by the TDC control law. Also plotted in Figures 4.4 and 4.5 are the trajectories generated using the constant gains of the linear controller used to form the linear reference model and applied to the actual nonlinear

ATTACHMENT 2

aeroelastic oscillator (labeled Linear Position / Velocity). Errors between trajectories under this linear control and the reference trajectory are due primarily to the nonlinear aerodynamic lift force. These plots show the degree of performance improvement (relative to a linear feedback law) that is possible with an adaptive controller operating under ideal conditions.

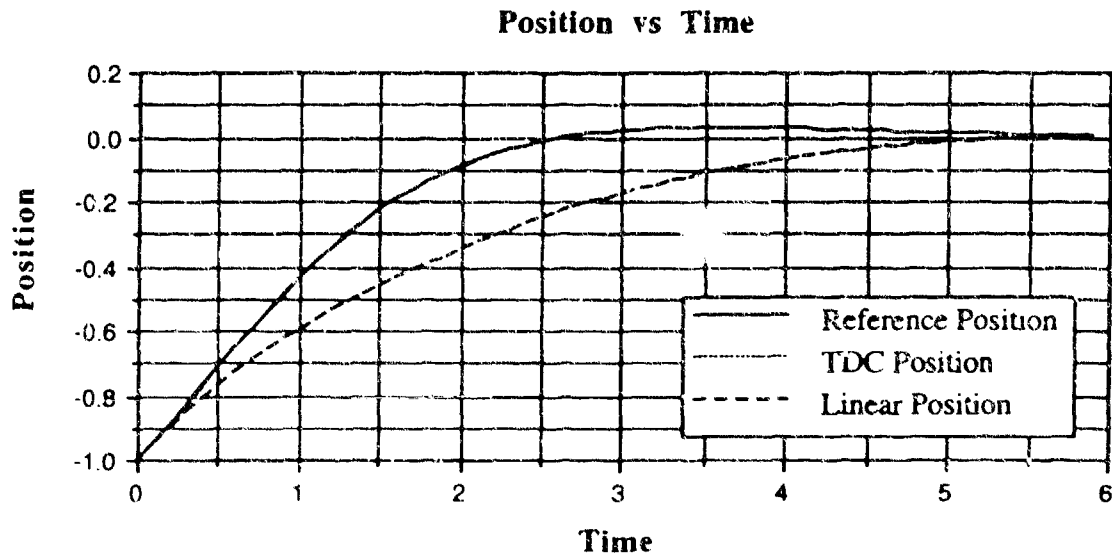


Figure 4.4 Position trajectories for the reference model, linear control law, and TDC at 200 Hertz controller rate.

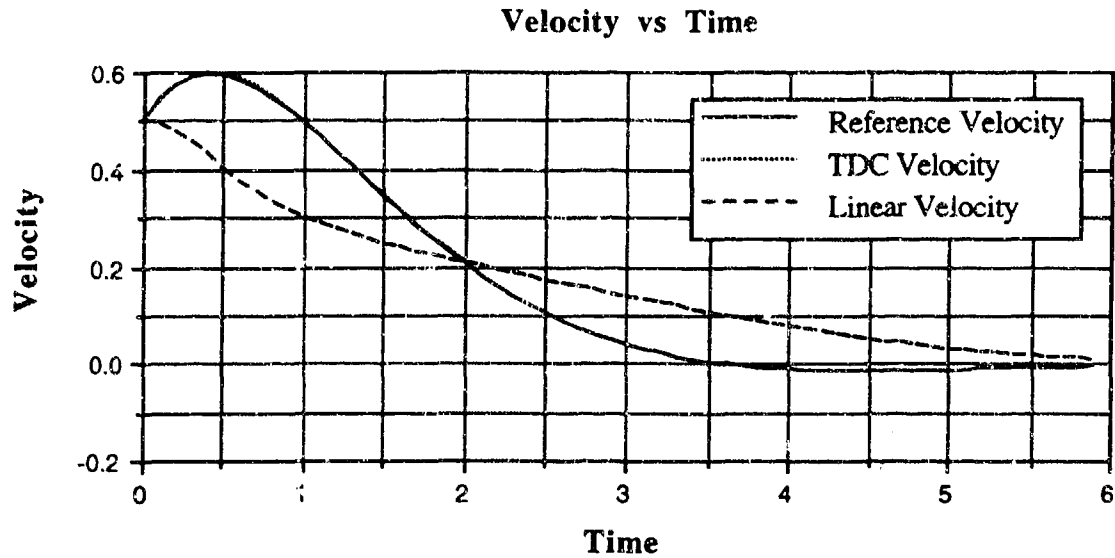


Figure 4.5 Velocity trajectories for the reference model, linear control law, and TDC at 200 Hertz controller rate.

Since the sensing and computational requirements associated with generating state information for the aeroelastic oscillator at the 200 Hertz integration rate may be unrealistic, the controller is slowed to calculate the control signal at a more moderate rate. For this experiment, the control was generated at 10 hertz. In order to produce unknown dynamics that are a function of control as well as state, an unknown external force equal to three times the control force was added to the unknown dynamics. In other words, the control form in Equation (4.7) was modified from the assumed known value

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} F$$

to the applied value

$$\left[\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 3 \end{bmatrix} \right] F$$

where the added term is not known by the controller. A relatively large force error was used to highlight the ability of the hybrid control system to reduce large uncertainties.

Consistent with the hybrid control law developed in Section 3.3, the learning

ATTACHMENT 2

system used a spatially localized network with 32 linear-Gaussian nodes. For training the network, a learning rate of 1 was used with the spatial decay of all the nodes fixed at 2. These values were selected based on the known mapping of the nonlinearities, the size of the input space (i.e., range of all possible position, velocity, and control combinations), and to a certain extent on trial and error. Initial values for the slopes and biases were set to zero while the Gaussian centers were placed randomly in the unit cube formed by scaling the state and control inputs. Figures 4.6 and 4.7 illustrate the hybrid controlled states for the first learning trial compared to the TDC controlled states and reference model. Since the slopes and biases of the learning system are initialized to zero, the learning system does not impact the states at start-up, and all of the unknown dynamics are incorporated into the TDC adaptive component. However, after a short time (within the first trial), the learning system begins to build a mapping of the unknown dynamics. This mapping is used to eliminate the delay associated with the unknown dynamics estimate in TDC and to improve the estimate of the local linearized behavior (i.e., using the derivative information as discussed in Section 3.3 to reduce model uncertainty). These features can be directly related to the improved performance seen in the state tracking of the reference trajectory.

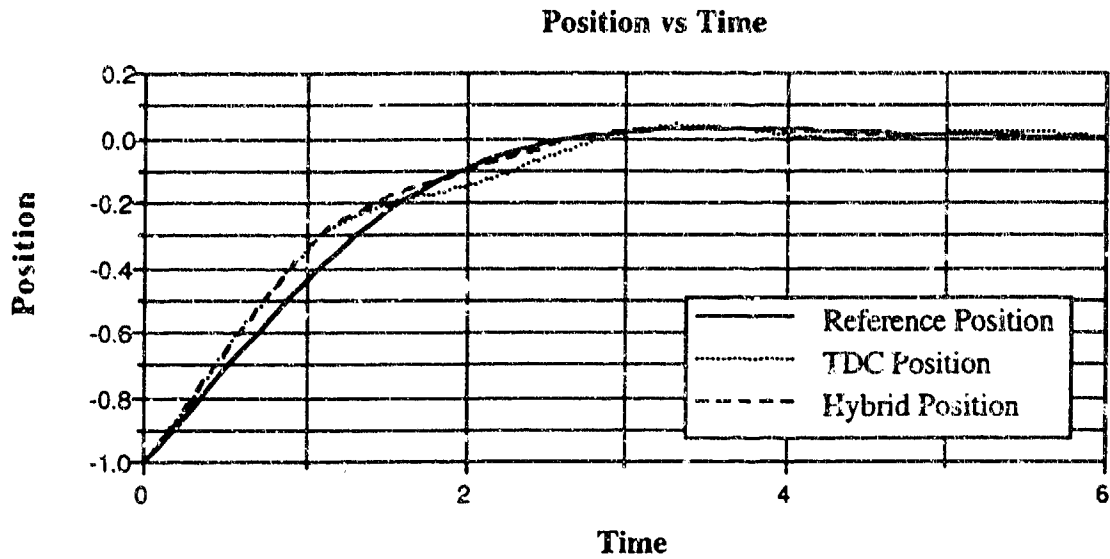


Figure 4.6 Position trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, first learning trial.

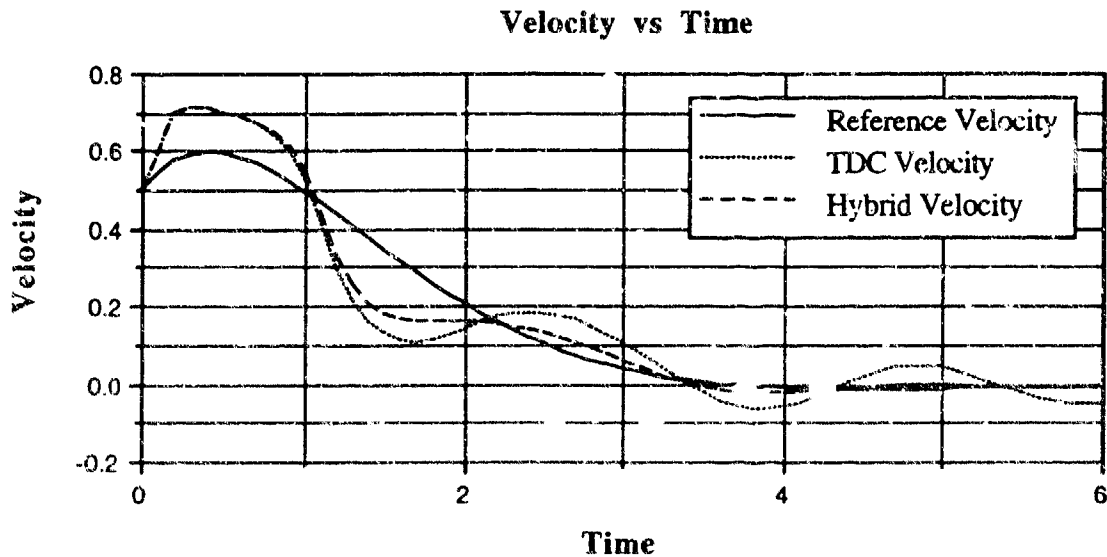


Figure 4.7 Velocity trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, first learning trial.

After the trajectory is repeated 10 times, the learning system has built a mapping of the previously unknown dynamics as a function of the state and control along that trajectory. Figure 4.8 compares the estimate of the unknown dynamics used by TDC to

ATTACHMENT 2

that of the hybrid controller generated from the learned mapping (after 10 trials). Since the mapping used to generate these points represents a static function, the unknown dynamics can simply be looked up as a function of the current state and control. This can be contrasted with TDC which uses an estimate of the unknown dynamics based on the state and control at the previous time.

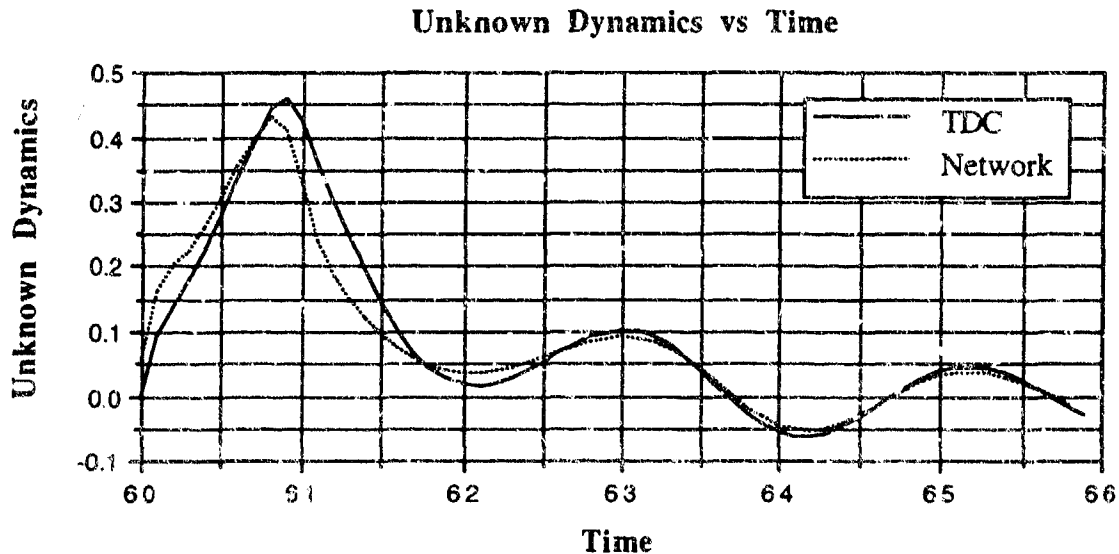


Figure 4.8 Unknown dynamics estimate from network and TDC after 10 trials.

As discussed in Section 3.3, the hybrid controller uses the output of the network (f_{net}) as well as the derivative of the network output with respect to the control ($\partial f_{net}/\partial u$) to formulate the control law. This derivative information provides local improvements to the linear control weighting vector, Γ . Since the truth model for the aeroelastic oscillator is known, it is possible to analyze the accuracy of the derivative information. For example, the partial of the unknown dynamics with respect to the control force is simply, in continuous time, the constant three (due to the added external control force). When converted to discrete time, this value is 0.3182. After 10 trials, the networks mean value of the $\partial f_{net}/\partial u$ is 0.2285. Although it has not yet learned the correct value, it nonetheless provides some improvement to the control weighing matrix.

ATTACHMENT 2

Figures 4.9 and 4.10 illustrate the state trajectories controlled by the TDC controller and the hybrid controller after 10 trials. Clearly, the hybrid controller uses experientially gained knowledge to improve the tracking of the reference states.

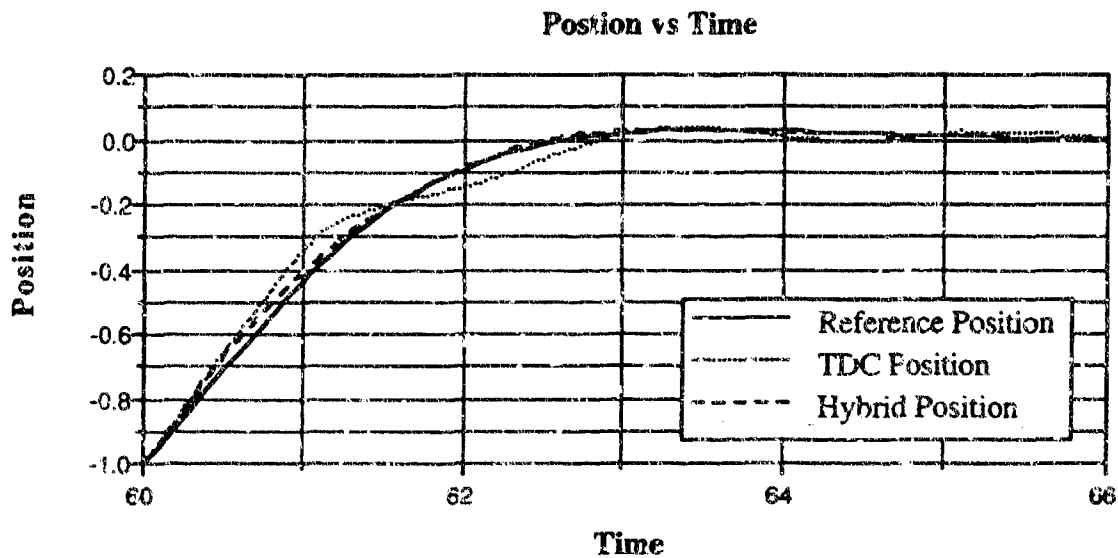


Figure 4.9 Position trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

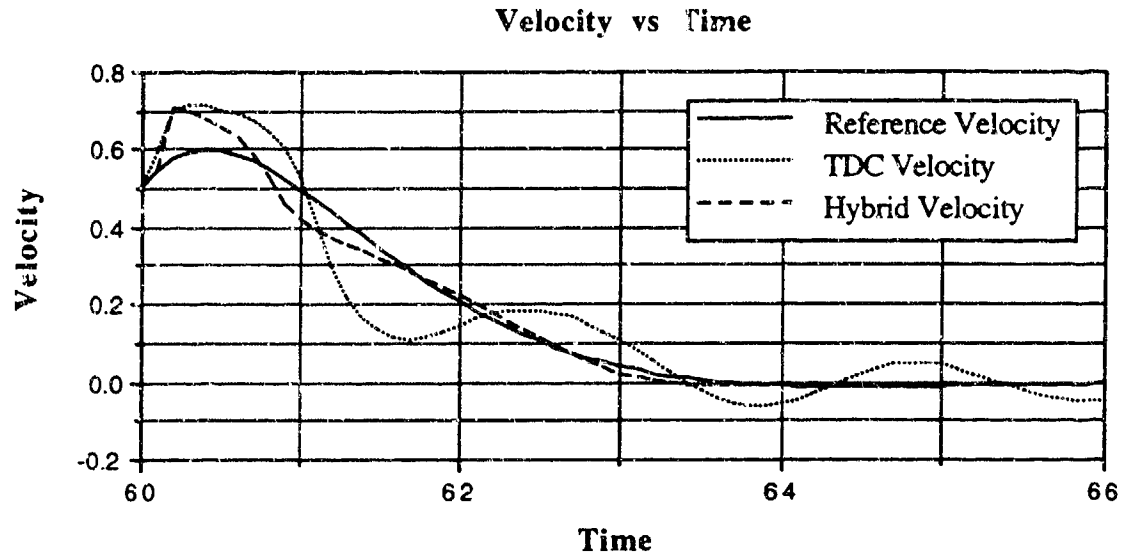


Figure 4.10 Velocity trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

This experiment shows that the hybrid controller has the ability to improve the controlled performance of the aeroelastic oscillator when a specific trajectory is repeated numerous times. This improved performance is realized by exploiting a learned functional mapping of the previously unknown model dynamics to improve the control law. The next experiment illustrates the ability to synthesize a mapping over a much larger input space, using randomly generated state trajectories.

4.1.6 Aeroelastic Oscillator Experiment 2

In experiment 2, the desired trajectory is selected in a random fashion in order to map the unknown dynamics over a much larger region of the state space than the single trajectory in experiment 1. By commanding a random position between -1 and 1 , a large portion of the state space along with the associated controls is visited and subsequently mapped. As in experiment 1, the aeroelastic oscillator was integrated at 200 Hertz and the control signal issued once every 20 integrations (10 hertz). For this experiment, a spatially localized learning system with 99 linear-Gaussian nodes was used. The spatial decay for

ATTACHMENT 2

each node was fixed to 1 and the initialization was the same as for experiment 1. The number of nodes, spatial decay, and other parameters were again selected based on the expected nonlinearities, size of the input space and trial and error.

Figure 4.11(a) shows the mapping synthesized by the learning system as a function of velocity and control. Learning was based on following the randomly generated reference trajectory for 60 seconds (10 trials). This mapping is compared to the nonlinear terms and extraneous control of the aeroelastic oscillator truth model shown in Figure 4.11(b). Comparing the two plots, the slope in the control direction (force) for the network mapping is nearly constant with a mean of 0.3120 and standard deviation of 0.0264 whereas the actual slope is 0.3182 (in the discrete time model). Moreover, the mappings in the velocity direction appear very similar. Hence, the network has synthesized the previously unknown dynamics of the system. (Note: the current version of the software does not allow a direct error surface plot to be generated.)

ATTACHMENT 2

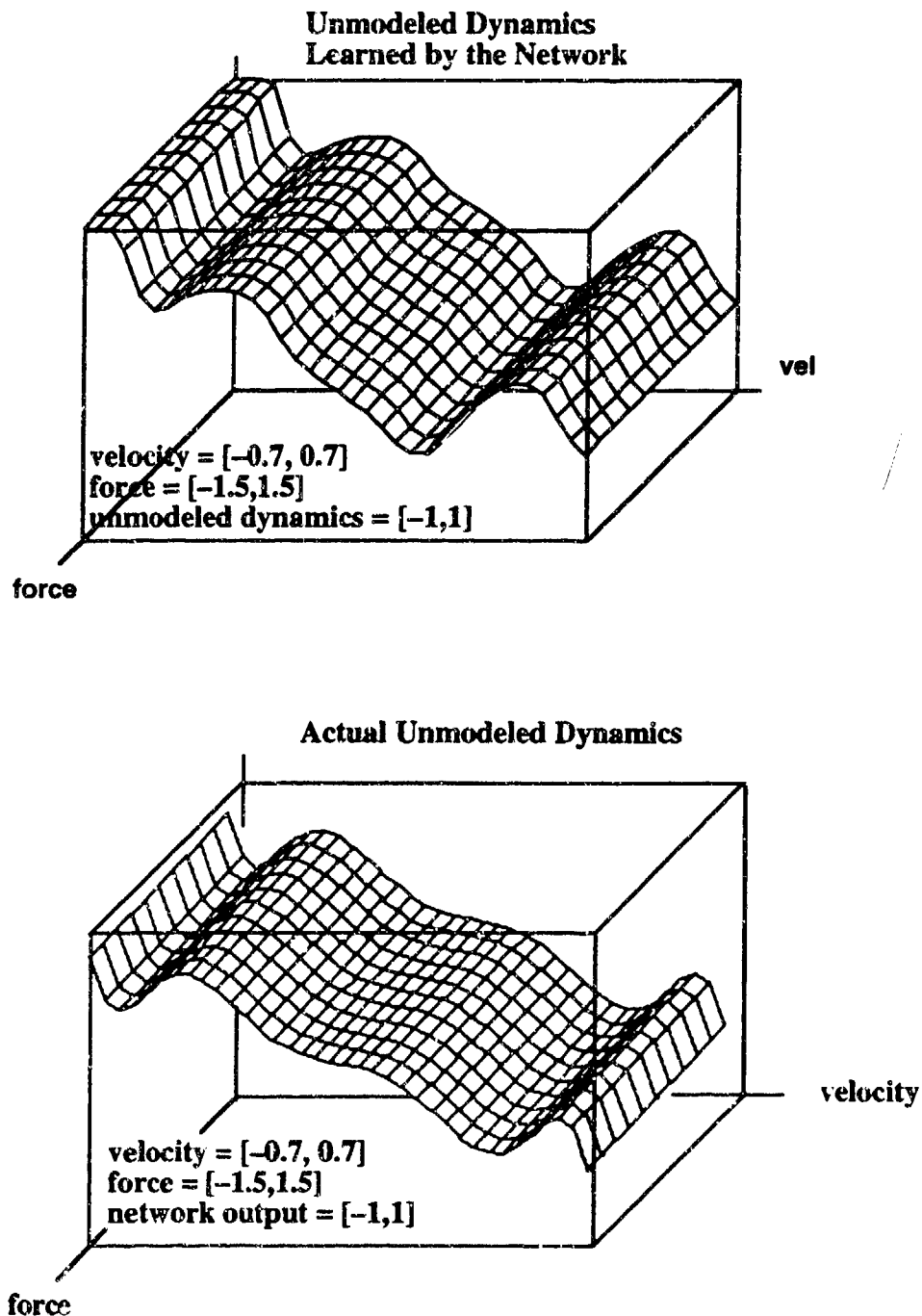


Figure 4.11 (a) Network Mapping of Unknown Dynamics (b) Actual Unknown Dynamics.

Figures 4.12 and 4.13 illustrate the position and velocity trajectories for the TDC and hybrid controlled states after 30 seconds of simulation. As predicted by the relatively accurate mapping of the unknown dynamics, the position and velocity show improved

ATTACHMENT 2

performance for the hybrid controlled aeroelastic oscillator over that of TDC.

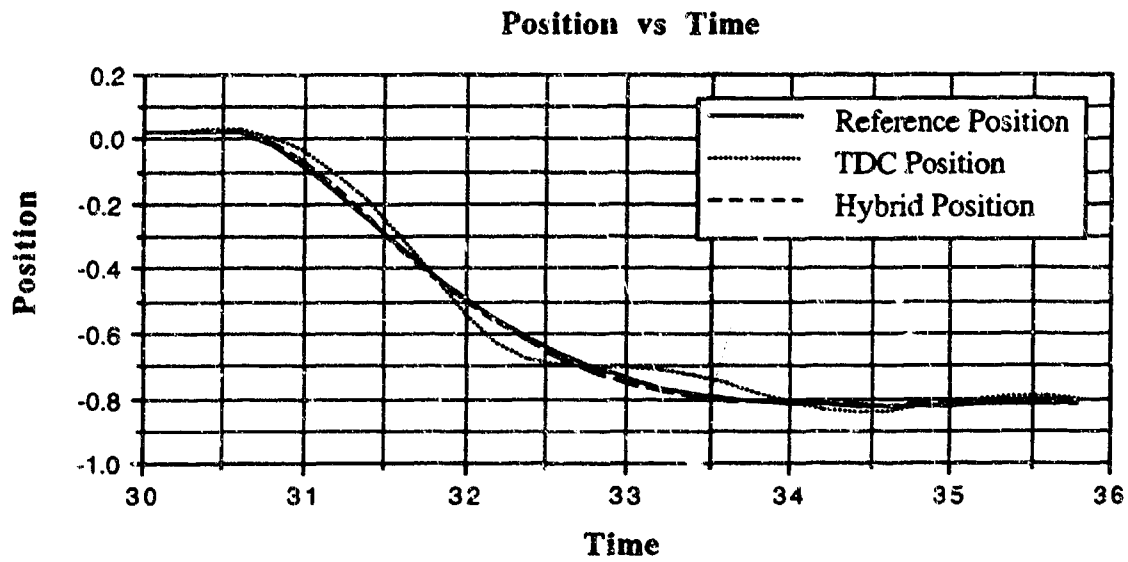


Figure 4.12 Position trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

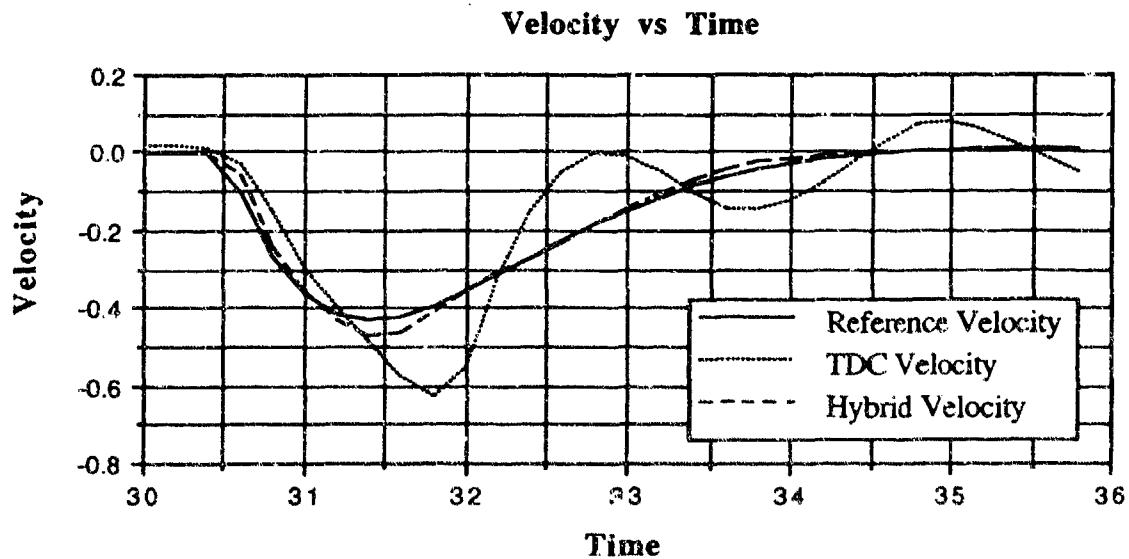


Figure 4.13 Velocity trajectories for the reference model, TDC, and hybrid control law at 10 Hertz controller rate, 10 trials.

4.2 HIGH PERFORMANCE AIRCRAFT MODEL

4.2.1 Aircraft Description

The high performance aircraft model that is used to illustrate the concept of learning enhanced flight control was developed by NASA to provide the aeronautical community with a common focus for research in flight control theory and design. This model is also being used to serve as a basis for the 1992 AIAA Control Design Challenge (Duke (1992)). A complete description of this generic, high performance, state-of-the-art aircraft model is found in Brumbaugh (1991). The following summarized the major characteristics of the aircraft model as well as its critical components.

The NASA model is the basis for the simulation of a high-performance, supersonic vehicle representative of modern fighter and attack aircraft. This model supports virtually all missions in nonterminal flight phases. These missions include flight phases that are normally accomplished using gradual maneuvers such as climb, cruise, or loiter as well as phases that require rapid maneuvering, precision tracking, or precise flight-path control (e.g., air-to-air combat, weapon delivery, or terrain following). The aircraft model includes full-envelope, nonlinear aerodynamics in addition to a full-envelope, nonlinear thrust model. An illustration of the basic configuration of the aircraft is shown below in Figure 4.14. Significant features of this aircraft configuration include a single vertical tail with rudder surface, a horizontal stabilator capable of symmetric and differential movement, and conventional trailing edge ailerons.

ATTACHMENT 2

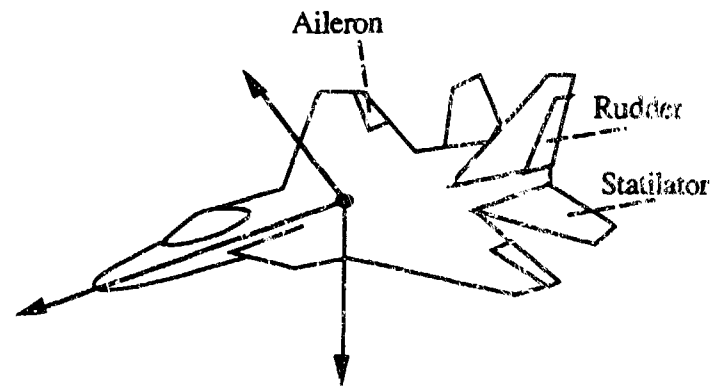


Figure 4.14 NASA High Performance Aircraft Model

The basic geometry and mass properties of the aircraft are summarized below in Table 4.2.

Table 4.2 Basic Aircraft Geometry and Mass Properties

| Aircraft Geometry and Mass Properties | |
|---------------------------------------|------------------------|
| Wing Area | 608.00 ft ² |
| Wing Span | 42.80 ft |
| Mean Chord | 15.95 ft |
| Weight | 45000.00 lb |

To aide in the design and development of a competent flight control law, the model can be easily broken into separate components, each performing a specific function. The major components of the aircraft model are as follows: aerodynamics, propulsion, actuator dynamics, and equations-of-motion. Also included with the model is the standard atmosphere component, an environmental model, and the integration component that is used to simulate the aircraft in software. Of course, one element that is not in this list is the flight control law, which is to be determined by the designer. The function of each of the major components, as well as a brief discussion of its origins, are presented in the following paragraphs.

As alluded to previously, the NASA aircraft simulation contains a nonlinear, full-

ATTACHMENT 2

envelope aerodynamic model. The primary function of this component is the calculation of aerodynamic forces and moments generated by the aircraft throughout its flight regime. In general, the aerodynamic forces and moments are complicated, nonlinear functions of many variables. The approach taken by the NASA model in calculating the complex force values is based on modeling the force terms as the product of dynamic pressure, a reference area (wing area), and an appropriate dimensionless aerodynamic coefficient. Similarly, the aerodynamics moment term is modeled as the product of dynamic pressure, a reference area, a dimensionless aerodynamic coefficient, and a reference length (mean chord). The aerodynamic coefficients are primarily functions of Mach number, angle-of-attack, and sideslip angle. The NASA aircraft model acquires coefficient values from multidimensional data tables or from direct calculation. The coefficients contained in the tabular data have been generated through a combination of wind-tunnel tests and computer programs that numerically integrate the theoretical aerodynamic pressure over the surface of the aircraft. For the tabular data, linear interpolation is employed to obtain intermediate values.

Vehicle thrust is generated by the *propulsion model*. Twin afterburning turbofan engines, each capable of generating 32,000 pounds of thrust, deliver power to the aircraft. Each engine thrust vector acts along the aircraft x-body axis at a point 10 feet behind the center of gravity and 4 feet laterally from the centerline. Engine dynamics are modeled by separating the powerplant into two separate sections. The first section, the engine core, is modeled as a first-order, closed-loop system that outputs thrust for a given throttle input. Moreover, rate limits that simulates spool-up effects and a gain scheduler that models changes in performance due to Mach number and altitude changes have been added to provide realism to the closed-loop system. Gains are obtained from tabular data and a linear interpolation routine based on Mach number and altitude. A second section, the afterburner, is modeled with similar first-order dynamics but has the added features of a rate limiter and sequencing logic to model fuel pump and pressure regulator effects. Together, these components comprise the full-envelope, nonlinear thrust model.

ATTACHMENT 2

Atmospheric parameters required by the aircraft simulation are computed by the *standard atmosphere model*. For a given altitude, values for acceleration due to gravity, speed of sound, temperature, and other essential parameters are generated from tables based on the U.S. Standard Atmosphere of 1962. Linear interpolation is used between elements of the table.

The *actuator dynamics model* is a first-order system that outputs surface position for a given surface command. Furthermore, rate and position limits are included in the system. All actuators are considered to be identical.

The dynamics of the aircraft are simulated using the *equations-of-motion module*. The nonlinear equations-of-motion are derived from the general six-degree-of-freedom relations for a rigid aircraft. Beyond the rigid body assumption, it is also assumed that the vehicle is traveling with nonzero forward motion in an atmosphere that is stationary with respect to an Earth-fixed reference frame. The nonzero forward motion assumption mandates that only nonterminal flight phases be simulated by this model. Since each degree of freedom requires two state variables (the basic variable and its rate), a total of twelve first-order differential equations are required to completely describe the motion of the aircraft. Table 4.3 lists each state variable and its symbol. Note that if speed is assumed to be relatively constant, then angle-of-attack and sideslip angle may be supplemented for the y and z body axis velocity vector projections respectively. A detailed derivation of these equations-of-motion can be found in Etkin (1982) or Roskam (1979).

The state variables are propagated in time via the *integration module*. This module uses a second-order Runge-Kutta midpoint algorithm to arrive at a new state based on the state and control at the previous time. Running at 50 Hertz, this integration technique has been found to provide a balanced tradeoff between numerical stability and processing speed.

ATTACHMENT 2

Table 4.3 The twelve aircraft state variables and symbols

| State Variables and Symbols | |
|-----------------------------|----------|
| Displacement North | x |
| Displacement East | y |
| Altitude | h |
| Velocity | u |
| Angle of Attack | α |
| Side Slip Angle | β |
| Pitch Angle | θ |
| Roll Angle | ϕ |
| Yaw Angle | ψ |
| Roll Rate | p |
| Pitch Rate | q |
| Yaw Rate | r |

An auxiliary component of the aircraft model that is not critical to the simulation but is invaluable to the control law designer is the *observation model*. The function of the observation model is to output a large class of aircraft measurements. States, state derivatives, accelerations, airdata parameters, force parameters, and a multitude of other important data are furnished for observation. Of these parameters, state information as well as vehicle body axis rates make up the set of parameters that have been traditionally used for feedback flight control.

By linking the previously described modules, a realistic, highly complex, nonlinear aircraft model that poses formidable challenges to the flight control designer is assembled. The high performance aircraft computer model received from NASA was written in the FORTRAN programming language. In order to produce a model compatible with the NetSim simulation and design package discussed in Section 4.1.4, this FORTRAN version

ATTACHMENT 2

was transposed into the C programming language by the author.

4.2.2 General Aircraft Characteristics

The flight envelope of the NASA aircraft model is characteristic of a high performance fighter aircraft (Brumbaugh (1991)). Figure 4.15 below illustrates approximate bounds of aircraft operation in terms of altitude and Mach number.

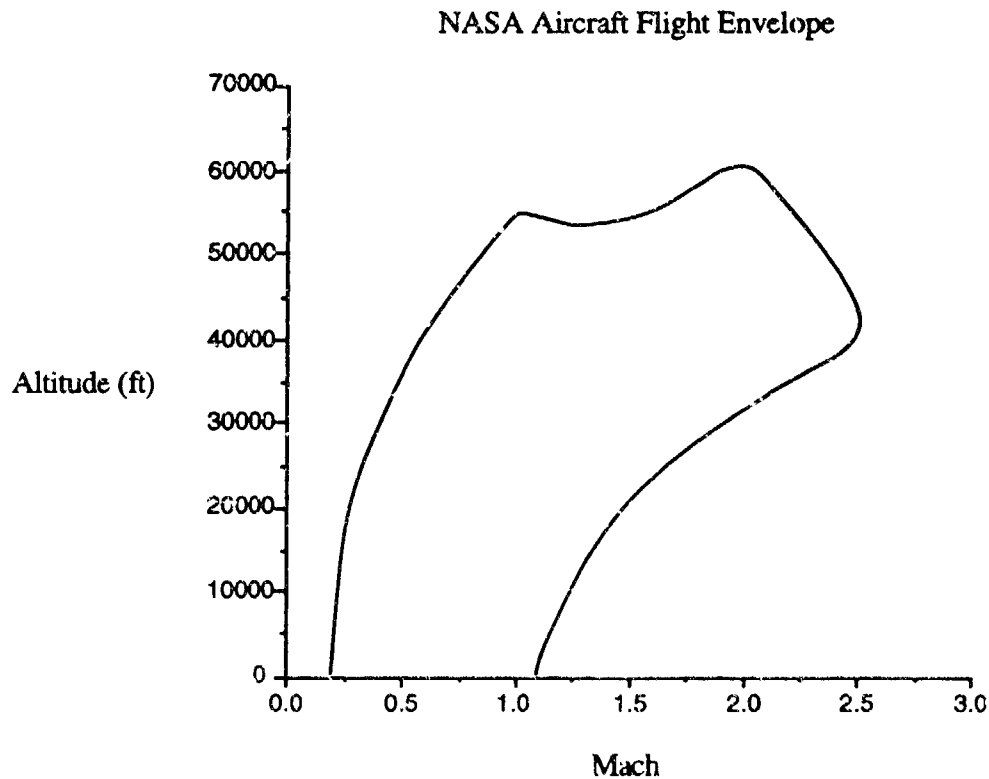


Figure 4.15 1-g Aircraft Flight Envelope.

To examine the nonlinear dynamics of a complex aircraft model, the equations-of-motion are frequently linearized about various operating conditions. By linearizing the dynamics at a sufficient number of operating conditions within the envelope, an improved overall picture of the actual nonlinear dynamics can be gained. Generally, operating conditions near the boundary of the envelope, as well as a few centrally located points, are

selected. The most common technique in obtaining linearized dynamics is by invoking the small perturbation theory based on a Taylor series expansion. This theory uses infinitesimal perturbations from an equilibrium or trimmed steady-state reference condition to predict aircraft response to perturbations that are not infinitesimal. A trim condition is classically defined as a constant velocity and altitude state with control surfaces and throttles set to maintain this condition. If it is assumed that all perturbations and their derivatives are small, the quadratic and higher order products of the perturbations will be negligible compared to the first-order quantities. In other words, a linear model is obtained by deriving relations of small deviations of all state and control variables about a steady-state equilibrium condition and retaining linear terms while ignoring quadratic and higher-order terms. A detailed version of the following short derivation of this theory can be found in (Athans (1990)).

Let $\mathbf{x}(t)$ and $\mathbf{u}(t)$ represent state and control variables, respectively, with

$$\mathbf{x}(t) \in \mathcal{R}^n \quad (4.8)$$

$$\mathbf{u}(t) \in \mathcal{R}^m \quad (4.9)$$

The nonlinear state dynamics in continuous time are given by

$$\dot{\mathbf{x}}(t) = \mathbf{f}\{\mathbf{x}(t), \mathbf{u}(t)\} \quad (4.10)$$

The reference state and control values representing an equilibrium condition (e.g., $\dot{\mathbf{x}}(t) = 0$) for the nonlinear equation are denoted by a subscript zero.

$$0 = \mathbf{f}\{\mathbf{x}_0, \mathbf{u}_0\} \quad (4.11)$$

Small perturbations about the equilibrium condition are denoted with a lower case delta:

$$\mathbf{x}(t) = \mathbf{x}_0 + \delta\mathbf{x}(t) \quad (4.12)$$

$$\dot{\mathbf{x}}(t) = \delta\dot{\mathbf{x}}(t) \quad (4.13)$$

$$\mathbf{u}(t) = \mathbf{u}_0 + \delta\mathbf{u}(t) \quad (4.14)$$

Expanding the state dynamics in a Taylor series about the equilibrium condition and solving

for $\delta \dot{\mathbf{x}}(t)$ while retaining linear terms and disregarding higher-order terms yields the following state perturbation dynamics:

$$\delta \dot{\mathbf{x}}(t) = \mathbf{A}_0 \delta \mathbf{x}(t) + \mathbf{B}_0 \delta \mathbf{u}(t) \quad (4.15)$$

where

$$(\mathbf{A}_0)_{ij} = \left. \frac{\partial f_i}{\partial x_j(t)} \right|_{\mathbf{x}(t)=\mathbf{x}_0, \mathbf{u}(t)=\mathbf{u}_0} \quad (4.16)$$

$$(\mathbf{B}_0)_{ij} = \left. \frac{\partial f_i}{\partial u_j(t)} \right|_{\mathbf{x}(t)=\mathbf{x}_0, \mathbf{u}(t)=\mathbf{u}_0} \quad (4.17)$$

\mathbf{A}_0 and \mathbf{B}_0 are the Jacobian matrices of the Taylor series expansion of $\mathbf{f}\{\mathbf{x}(t), \mathbf{u}(t)\}$ centered about \mathbf{x}_0 and \mathbf{u}_0 . Although the Jacobian matrices can occasionally be found in closed form for relatively simple systems, more complex systems often require numerical differentiation. For this reason, numerical differentiation is used for the aircraft model to calculate the Jacobian matrices.

Using the small perturbation theory to linearize the equations-of-motion about an equilibrium condition can provide insight into the local behavior of the nonlinear aircraft dynamics in terms of stability, transient responses, and other system characteristics. However, this theory is not without its limitations. Large numbers of linear models must be computed to characterize the dynamics in highly nonlinear regions of the flight envelope. Moreover, the small perturbation theory is ill-suited to handle phases of flight where large deviations from the nominal trim condition are encountered (i.e., high angle-of-attack flight or spinning maneuvers).

Linearizing the equations-of-motion of the NASA aircraft has revealed that the longitudinal dynamics are only lightly coupled with the lateral dynamics at the majority of flight conditions. Moreover, control of the longitudinal or pitching motion is dominated by symmetric movement of the horizontal tail and engine thrust whereas the rolling and yawing motions associated with the lateral dynamics are most heavily influenced by the

ailerons and differential movements of the horizontal tail. For this reason, the aircraft flight control design problem can be separated into two distinct problems, each less complex than the whole. The existence of the lightly coupled modes and the ability to decompose the control system design is common to all but the most unconventional aircraft.

The uncoupled, linearized longitudinal dynamics of the aircraft can be described by a total of five coupled linear, time-invariant differential equations that are a function of pitch rate, velocity, angle-of-attack, pitch angle, and altitude. If the linearized equation for the dynamics of the total thrust in the longitudinal direction is added to this set of variables, the state of the aircraft for longitudinal motion is as follows (where T is total thrust):

$$\mathbf{x} = [q \quad u \quad \alpha \quad \theta \quad h \quad T]^T \quad (4.18)$$

If the dynamics of the inertial altitude and thrust are temporarily neglected, the four remaining differential equations define the traditional natural modes associated with aircraft pitching motion, namely the short period and phugoid modes. The short period mode is characterized by a highly damped, high frequency oscillation. The short period oscillations represent changes in angle-of-attack and pitch angle with near constant trim speed. In contrast, the phugoid mode exhibits very lightly damped, low frequency oscillations when excited. Under the influence of the phugoid mode, the angle-of-attack remains essentially constant while the speed and pitch angle experience changes. This motion represents a continual exchange of kinetic and potential energy of a slowly rising and falling airplane. Table 4.4 contains the natural frequencies (ω_n) as well as the damping ratios (ξ) for the open-loop longitudinal modes (sp = short period, ph = phugoid) of the NASA aircraft model at four equilibrium points (trim conditions) near the subsonic boundary of the flight envelope. Trim condition 5, which is not on the boundary, is included since it will be used as the initial condition for experiments described in Sections 4.2.5 and 4.2.6.

ATTACHMENT 2

Table 4.4 High performance aircraft longitudinal modes at various altitude and Mach number trim conditions.

| Natural Frequency and Damping Ratio: Longitudinal Modes | | | | | | |
|---|----------|------|----------------|------------|----------------|------------|
| Trim Condition | Altitude | Mach | ω_{nsp} | ξ_{sp} | ω_{nph} | ξ_{ph} |
| 1 | 5000 | 0.31 | 1.88 | 0.58 | 0.11 | 0.04 |
| 2 | 5000 | 0.90 | 4.68 | 0.28 | ** | ** |
| 3 | 35000 | 0.68 | 1.92 | 0.32 | 0.08 | 0.10 |
| 4 | 35000 | 0.90 | 2.11 | 0.21 | 0.02 | 0.12 |
| 5 | 9800 | 0.60 | 2.77 | 0.52 | 0.08 | 0.07 |
| ** at this trim condition, the aircraft does not exhibit a phugoid motion | | | | | | |

For purposes of comparison, the values of natural frequency and damping ratio for a high maneuverability aircraft in nonterminal flight phases can be found in the military specification regulation, MIL-F-8785C (1980). This regulation requires the phugoid mode to have a damping ratio greater than 0.04 and the short period damping ratio to be between 0.35 and 1.30. Moreover, the short period must have a natural frequency approximately bounded by 1 and 10 radians per second, depending on load factor and angle-of-attack. Examining Table 4.4 above, the NASA aircraft fails to meet the requirements for longitudinal motion in some areas of the flight envelope. However, through the use of a control system, the aircraft modes can be modified to meet the military specifications. For the hybrid control law, this is accomplished by selecting a reference model that meets these specifications.

4.3.3 Aircraft Reference Model

As discussed in Section 3.3, the reference model generates the desired state trajectory for the hybrid controlled aircraft states. During the process of selecting a reference model, close attention was paid to ensuring that following the reference trajectories did not require unrealistic control actions. Since the rate and position of the

horizontal stabilator is limited, unrealistic demands on control can translate into either rate of position saturation (e.g., an inability to exercise the control that has been calculated by the control law). Control saturation leads to inadequate performance (i.e., fails military or other specifications) and possibly to instabilities.

The reference model for the high performance aircraft was chosen to be the linear closed-loop system that results from applying an optimal linear control design to the open-loop dynamics linearized about a selected trim condition. For the experiments in Section 4.2.5 and 4.2.6, the linearized dynamics at trim condition 5 (see Table 4.4) were used. The state and control weights for the quadratic cost function used by the optimal control law were initially selected using guidelines suggested by Bryson & Ho (1975) and Kwakernaak & Sivan (1972). Trial and error (based on simulations of the linear dynamics) were used to arrive at the final cost function. The natural frequency and damping ratios for the modes of the closed-loop reference system are listed in Table 4.5.

Table 4.5 Reference Model Longitudinal Modes

| Natural Frequency and Damping Ratio | | | |
|-------------------------------------|------------|----------------|------------|
| ω_{nsp} | ξ_{sp} | ω_{nph} | ξ_{ph} |
| 1.46 | 0.96 | 0.75 | 0.96 |

Compared to the open-loop dynamics, the closed-loop reference model has modes that are heavily damped. Moreover, the natural frequency of the phugoid is much higher in the closed-loop system. This reference system meets military specification requirements.

4.2.4 Application Issues

In this section, the application of the hybrid flight control law to the high performance aircraft model is discussed. Figure 4.16 illustrates the block diagram representing the closed-loop simulation of the hybrid controlled aircraft model in the NetSim simulation and design package.

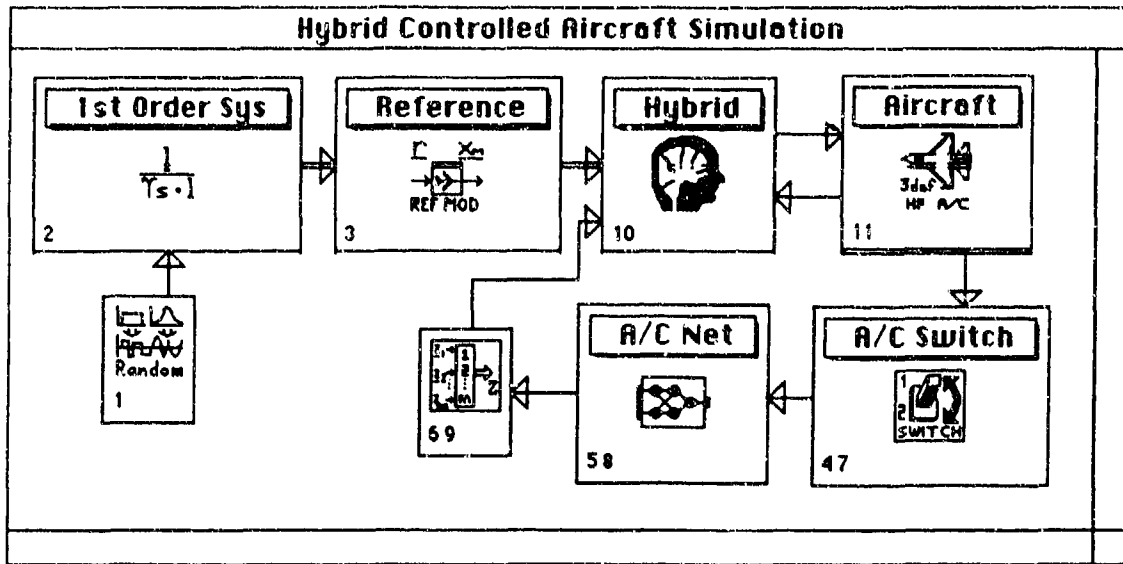


Figure 4.16 Block Diagram of the Hybrid Controlled Aircraft Simulation

The main modules in Figure 4.16 represent the reference model, hybrid controller, high performance aircraft model, and the linear-Gaussian network. The function of the remaining modules is to modify the output signals (represented by the connecting arrows) passed to the main modules. Again, the number in the lower left corner of each block dictates the order of execution at each time step. Modules that are called more than once per time step are shown with multiple sequence numbers. The following paragraph outlines the principal function of each module.

Random is the first module that is executed. It generates randomly selected reference commands for the altitude and velocity of the aircraft within a user-defined operating range. The length of time these commands are held constant before a new set of reference commands is generated is also determined by the user. The commands are supplied to the *1st Order Sys* module. This module processes the reference commands with a user-defined, rate-limited first-order filter. The purpose of this module is to smooth the step commands generated by the random module, effectively outputting a smoothed ramp to a step command. The function of *Reference* is to generate the desired state trajectory that is to be followed by the hybrid control law. The reference model that is used

is discussed in Section 4.2.1. The *A/C Switch* module supplies the state at the current time, as well as the state and control at the previous time step to the network. The switch also sends a flag to the network to ensure that learning only occurs with states and controls that are at consistent times. The linear-Gaussian network in the hybrid control architecture is contained in *A/C Net*. The role of the *Multiplexor* (shown with sequence numbers 6 and 9) is to store the output of the learned mapping for various inputs of state and control required by the hybrid control law. *Hybrid* executes the hybrid control law developed in Section 3.3. The complete high performance aircraft model is contained in the *Aircraft* module.

4.2.5 High Performance Aircraft Experiment 1

In experiment 1, the aircraft was given random commands for altitude and velocity. More specifically, the random altitude commands were between ± 500 feet and the random velocity commands were between ± 10 feet per second. As discussed in Section 4.2.4, the commands are filtered by a rate limited, first-order system. The rate limits for altitude and velocity were set to 50 feet per second and 4 feet per second per second, respectively. The filtering and rate limiting is intended to result in a physically feasible reference trajectory. The initial condition for the aircraft was an equilibrium condition at an altitude of 9800 feet and velocity of 539 feet per second (trim condition 5 in Table 4.4). For each new randomly generated command, the aircraft was reinitialized to this same trim condition. By randomly selecting commands, the objective was to generate state trajectories that fully traverse a small region of the aircraft operating envelope.

Similar to the experiments involving the aeroelastic oscillator, the linearized dynamics of the aircraft supplied to the hybrid controller were perturbed from their actual values. The purpose of the perturbations was to increase model uncertainty, a feature the hybrid controller is able to accommodate. The perturbations to the dynamics can be viewed as a situation wherein the flight control system is provided linearized dynamics that

ATTACHMENT 2

represent a trim condition other than that for which the maneuvers are actually to take place. The intent is to illustrate that the hybrid controller is able to adequately control the aircraft given an inaccurate linear representation, indicating that less accurate *a priori* design information is needed and thereby use of the hybrid controller can effectively reduce design costs.

The learning component used in the hybrid control law was again the spatially localized system developed in Section 3.2. For this case, the network consisted of 8 linear-Gaussian nodes. This relatively small number of nodes was considered to be sufficient due to the modest nonlinearities expected for the specified class of reference trajectories. Of the two largest factors in determining the nonlinearity of the system, angle-of-attack and Mach number, only angle-of-attack experiences significant changes during the maneuvers associated with these reference trajectories. This is due to the relatively small commanded changes in altitude and velocity when compared to the flight envelope, and thus small changes in Mach number. The centers of the linear-Gaussian nodes were arranged in a user-defined grid over the input space, with the highest density of nodes in the angle-of-attack dimension (due to expected nonlinearities). Moreover, the spatial decay of each node was varied as a function of the center location of its nearest neighbor. The closer the neighboring center, the higher the spatial decay, and conversely, the farther the neighboring center, the lower the spatial decay. This pattern ensures that each point in the input space can be adequately mapped to the desired output values. Initial values for the slopes and biases of the linear-Gaussian nodes were set to zero, since no *a priori* design information was assumed. Due to this initialization to zero, the learning system does not impact the states at start-up and all of the unknown dynamics are initially faced by the TDC adaptive component. After evaluating the relative magnitude of each element of the unknown dynamics and disturbance vectors supplied by the adaptive component, the cost function (Equation 3.20) was weighted to ensure all errors between the desired output and actual network output have the same significance. Equation (4.19) demonstrates how the cost

ATTACHMENT 2

function can be weighted for specific errors between the desired and actual network output:

$$J = \frac{1}{2} [\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x}, \mathbf{p})]^T \mathbf{C} [\mathbf{d}(\mathbf{x}) - \mathbf{f}_{net}(\mathbf{x}, \mathbf{p})] \quad (4.19)$$

where \mathbf{C} is a diagonal matrix with user-supplied weights along the diagonal. The global learning rate (α) was selected by trial and error in order to find the highest rate of convergence to the desired output with adequate accuracy while still maintaining a static mapping (i.e., one for which parameters are not in a continuous state of change).

The model of the aircraft dynamics, which is in a continuous time form, was integrated at 50 Hertz to provide a balanced tradeoff between numerical stability and processing speed. However, the control signal was calculated at a more moderate rate of 10 Hertz in order to reduce the real-time sensing and computation requirements in determining the complete state.

After running the simulation with randomly generated commands for 500 trials, of 20 seconds each, the learning system was able to build a mapping of a significant amount of the previously unknown dynamics. Since the true mapping of the unknown dynamics is not known (in contrast to the case with the aeroelastic oscillator), the cost, as defined in Equation (4.19), is used as a measure of performance of the learning system. Figure 4.17 illustrates both the initial cost and the cost after learning after 500 trials for a 500 foot climb and simultaneous 10 feet per second increase in velocity.

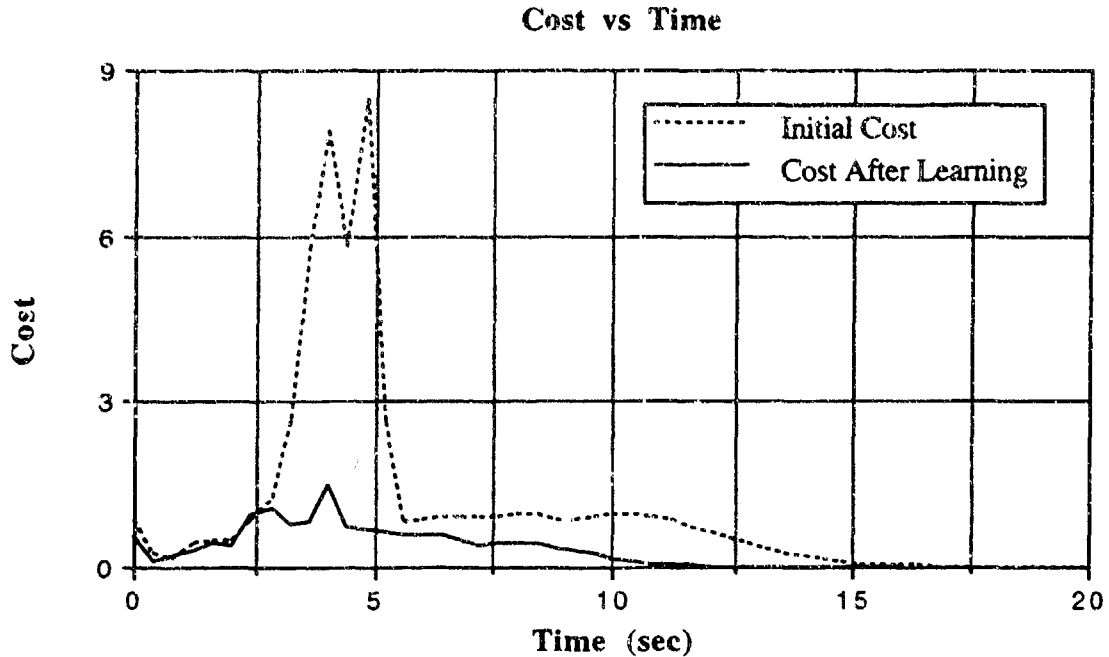


Figure 4.17 Comparison between initial cost and cost after 500 trials.

Since the cost is significantly less for the case after learning, this indicates that the learning component of the hybrid law has built a mapping of a significant amount of the unknown dynamics. If the simulation is allowed to run even longer, the cost will further decrease. However, since the true aircraft model dynamics are very high dimensional and contains states that are not included as inputs to the network (e.g., the state of the actuators), it is impossible to completely learn the initially unknown dynamics. For this reason, there will always be a finite, non-zero cost.

The state trajectories for the reference model, for the TDC controlled aircraft, and for the hybrid controlled aircraft for a commanded 500 foot climb and 10 feet per second increase in velocity are shown in Figures 4.18(a) through 4.23(a). Since the difference between these trajectories is typically small compared to the absolute initial trim values, the errors between the desired reference and the actual trajectory for both the TDC and hybrid controlled aircraft are shown in Figures 4.18(b) through 4.23(b). The horizontal stabilator deflection and throttle position for the TDC and hybrid controlled aircraft are shown in

ATTACHMENT #2

Figures 4.24 and 4.25. These values represent the actual values used on the aircraft. Due to actuator dynamics, these actual values are generally not the commanded output calculated by the given control law.

As illustrated by the state trajectories, the hybrid control law offers improvements over the TDC controller. Although the errors in velocity and altitude are relatively small, errors in the vehicle rates and angles are significant in the sense that oscillations about the reference trajectory are reduced. This reduction in oscillations for the hybrid controlled aircraft has the potential to change a response that was formerly objectionable to the pilot to one that is satisfactory. Moreover, the horizontal stabilator deflection for the hybrid controlled aircraft is improved over that of the TDC controlled aircraft in the sense that the control signal is less oscillatory (and subsequently less taxing on the actuators).

The trajectories for the reference model and the hybrid controlled aircraft differ for two major reasons. The first, as previously discussed, is the inability of the learning system to map the unknown dynamics for states that are not given as inputs to the network (e.g., actuator states). Perhaps more significant are the difficulties associated with attempting to control more states than there are available control inputs. Since a pseudo-inverse must be used in the hybrid control law when the number of controls is less than the number of states as discussed in Section 3.3, the tracking of the complete state is not guaranteed even for a simulated case without any unknown dynamics (Anderson & Schmidt (1990)). Due to this inability to control all the state variables, it is almost certain that differences will exist between the reference and actual trajectories. As a result, errors between the reference trajectory and the hybrid controlled trajectory do not necessarily represent a failure of the learning system to map the unknown dynamics, but an inability to control all the states to a reference trajectory with a limited number of controls.

ATTACHMENT 2

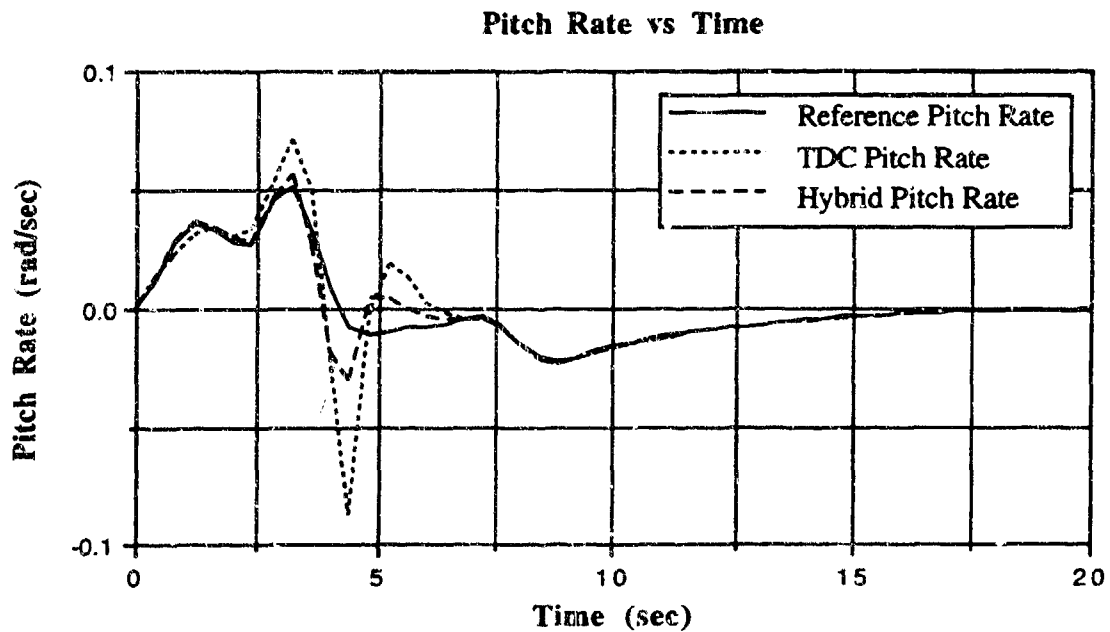


Figure 4.18(a) Pitch rate trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

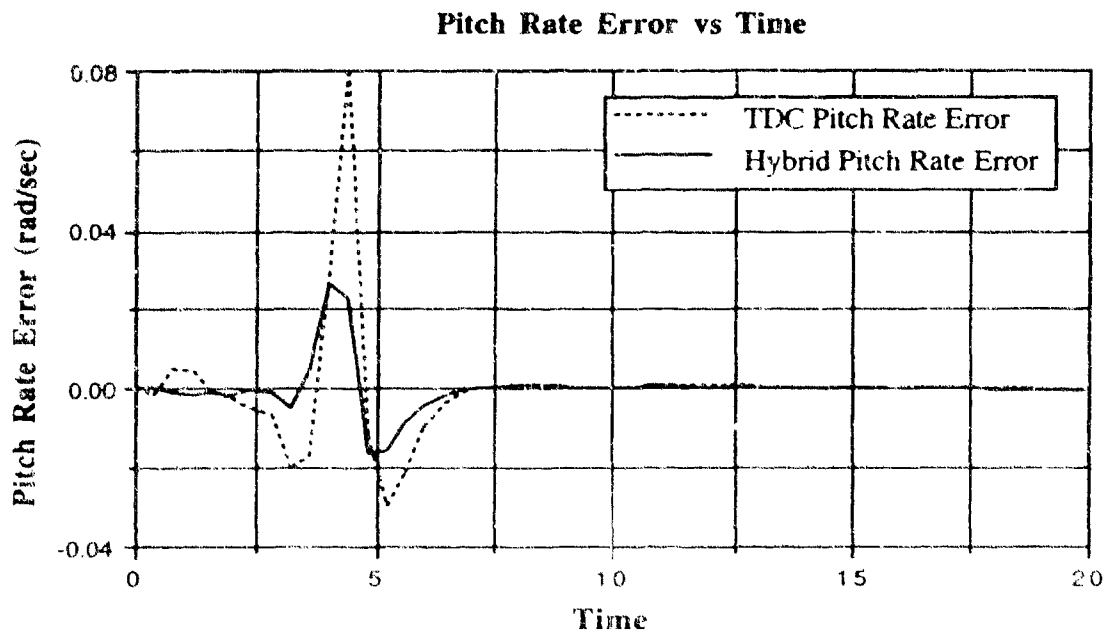


Figure 4.18(b) Error in pitch rate between reference trajectory and TDC or hybrid controlled aircraft.

ATTACHMENT 2

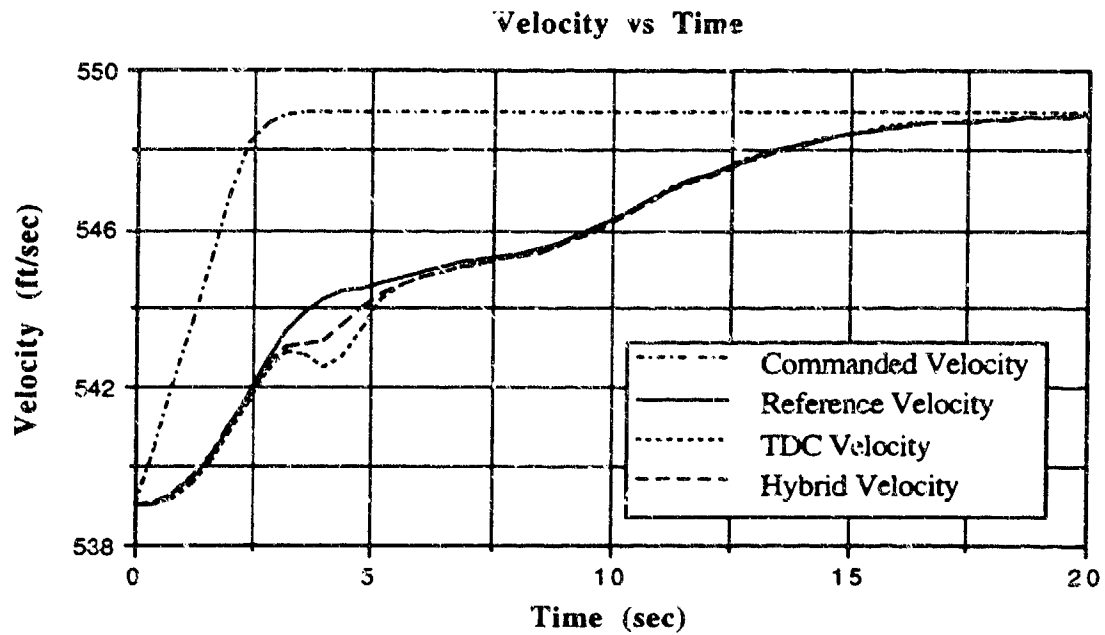


Figure 4.19(a) Velocity trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

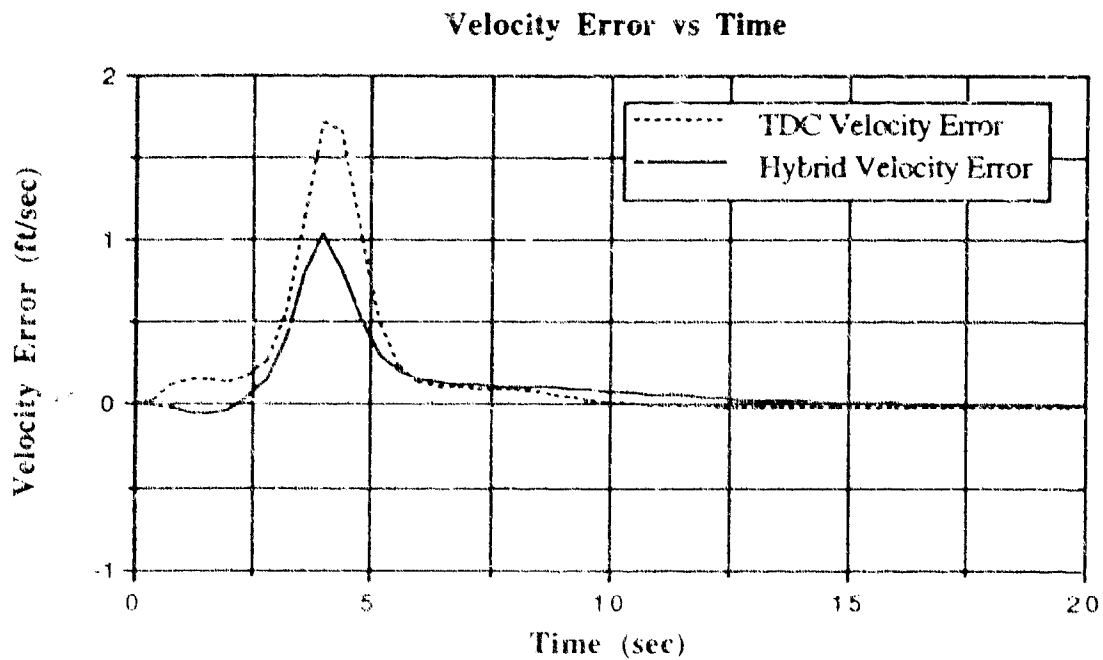


Figure 4.19(b) Error in velocity between reference trajectory and TDC or hybrid controlled aircraft.

ATTACHMENT 2

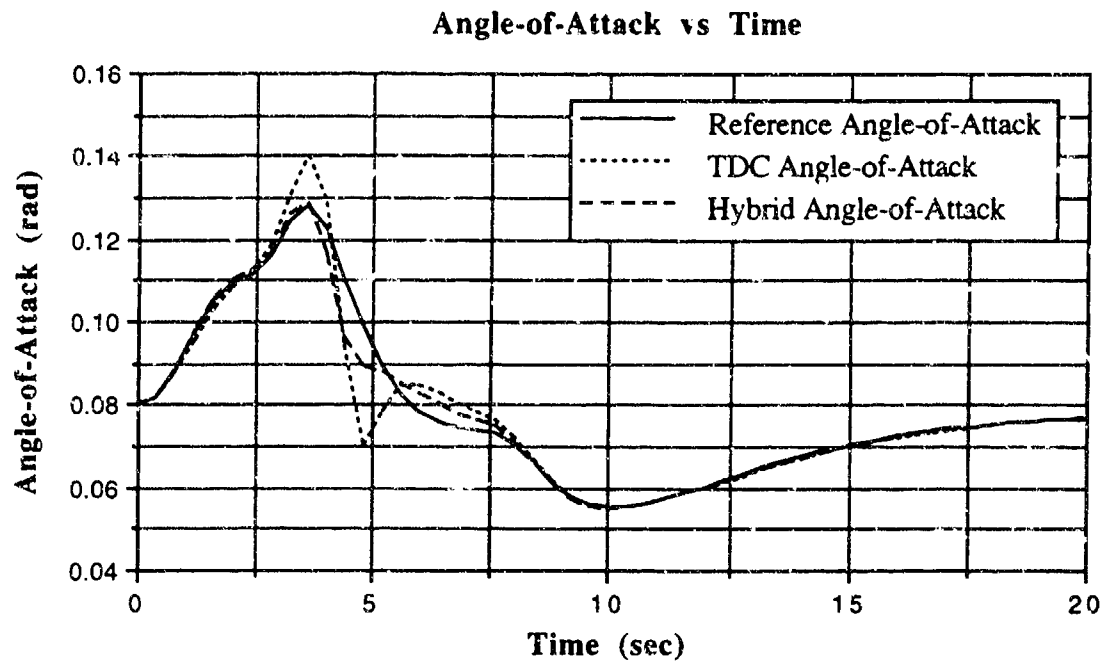


Figure 4.20(a) Angle-of-attack trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

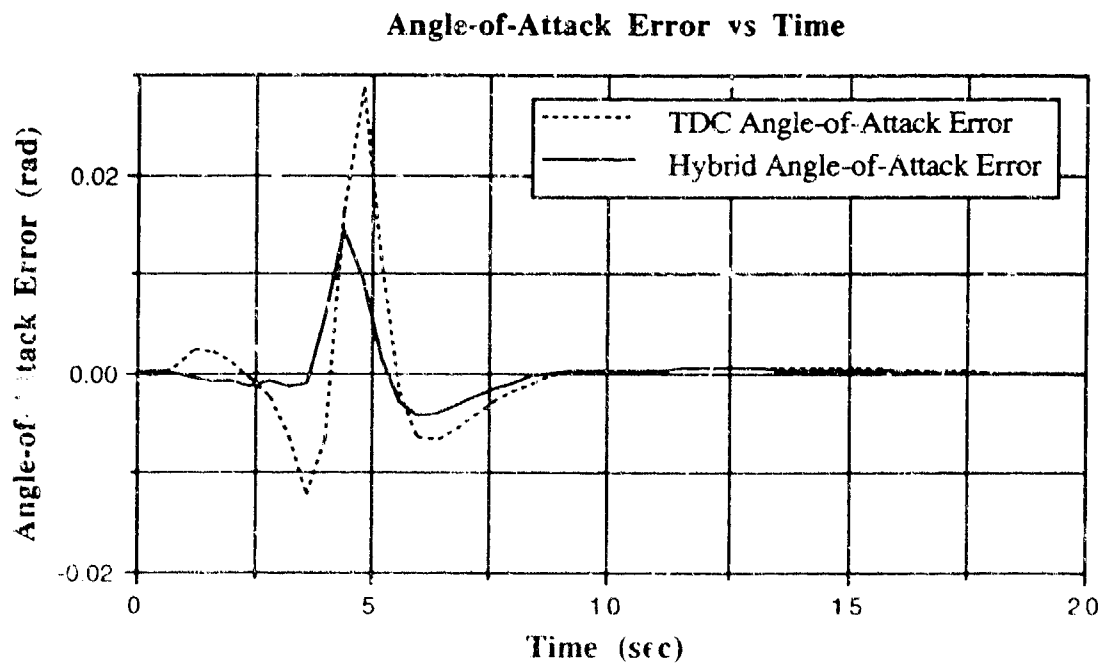


Figure 4.20(b) Error in angle-of-attack between reference trajectory and TDC or hybrid controlled aircraft.

ATTACHMENT 2

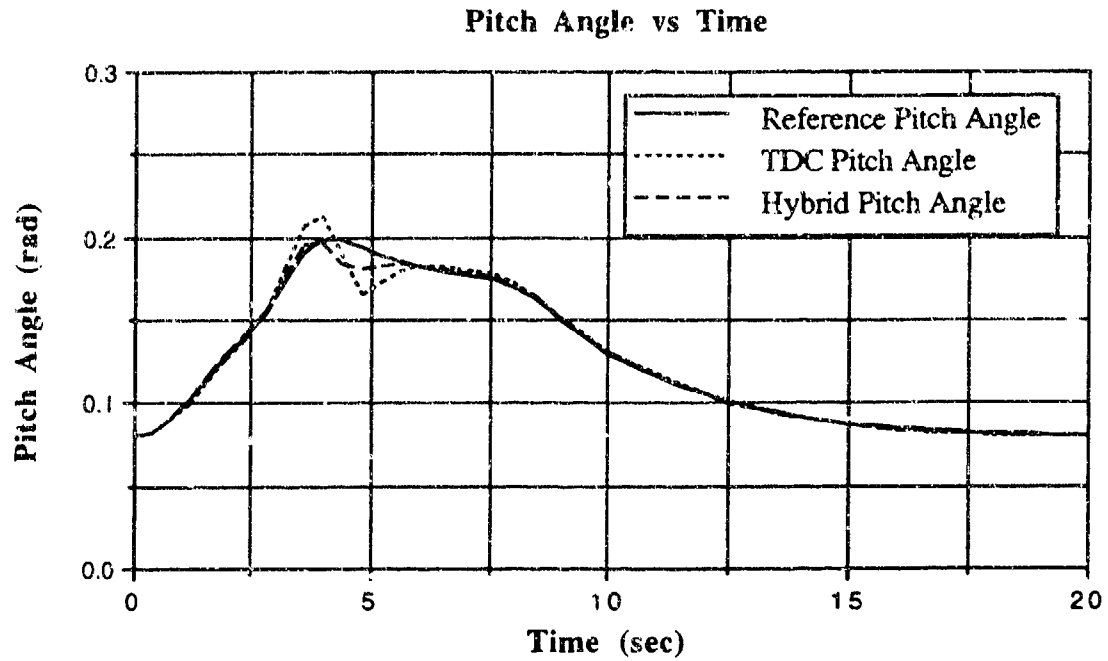


Figure 4.21(a) Pitch angle trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

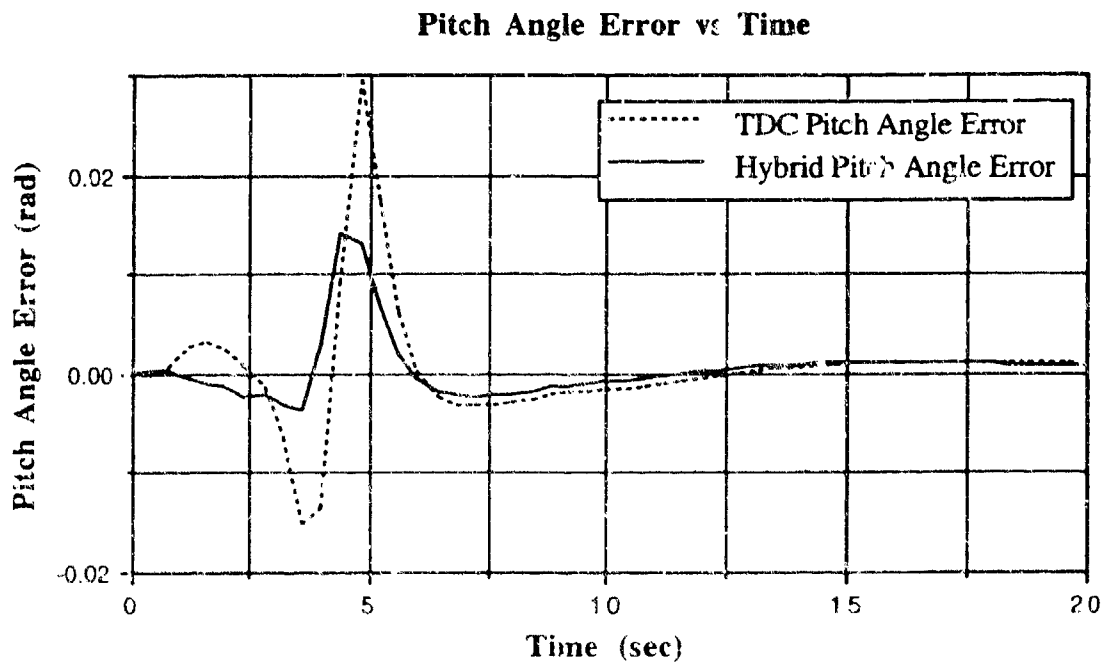


Figure 4.21(b) Error in pitch angle between reference trajectory and TDC or hybrid controlled aircraft.

ATTACHMENT 2

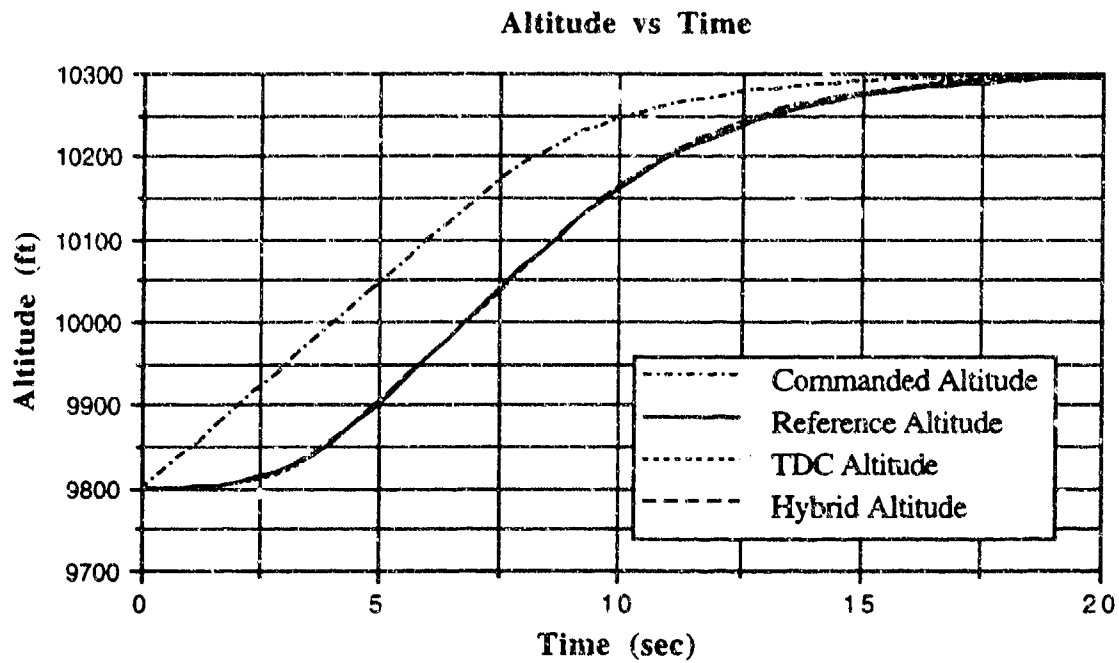


Figure 4.22(a) Altitude trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

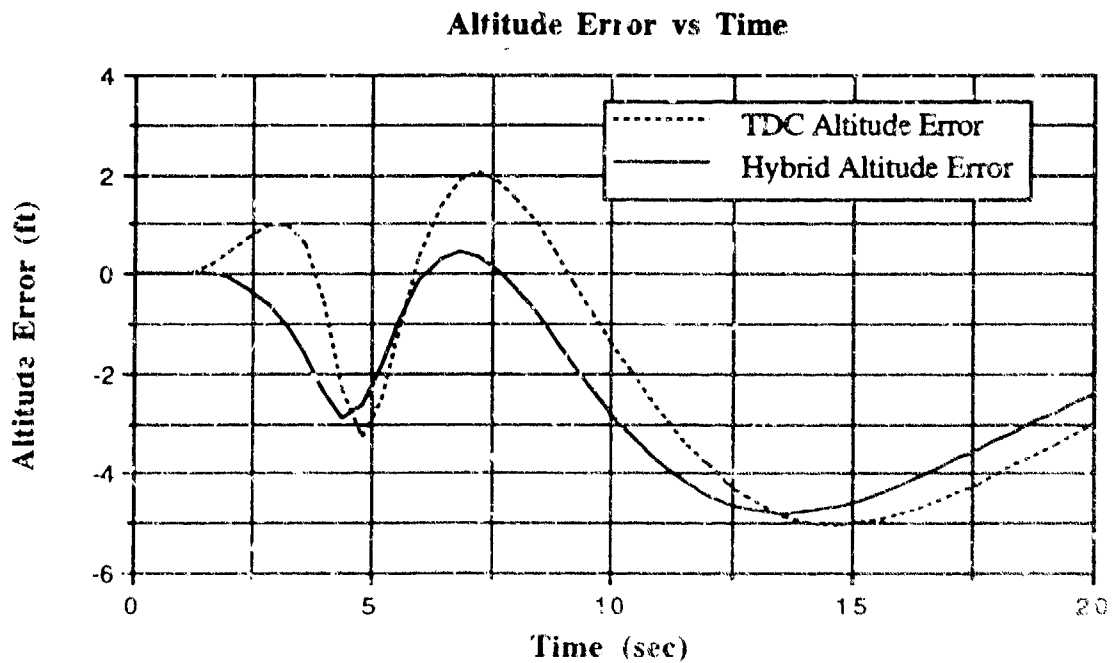


Figure 4.22(b) Error in altitude between reference trajectory and TDC or hybrid controlled aircraft.

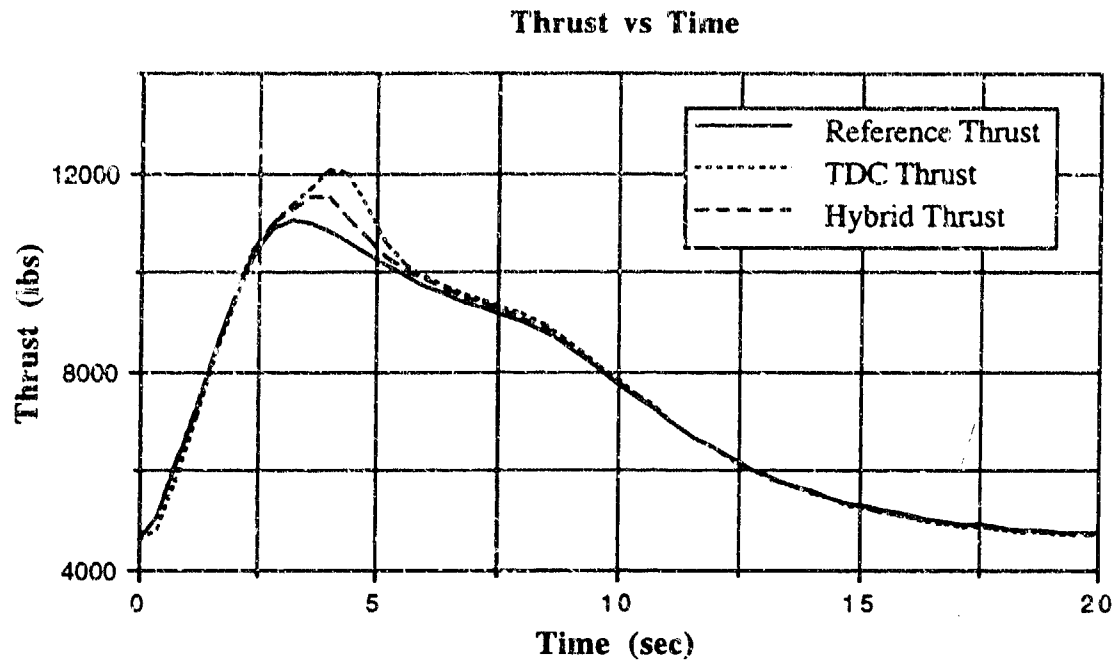


Figure 4.23(a) Thrust trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft after 500 trials.

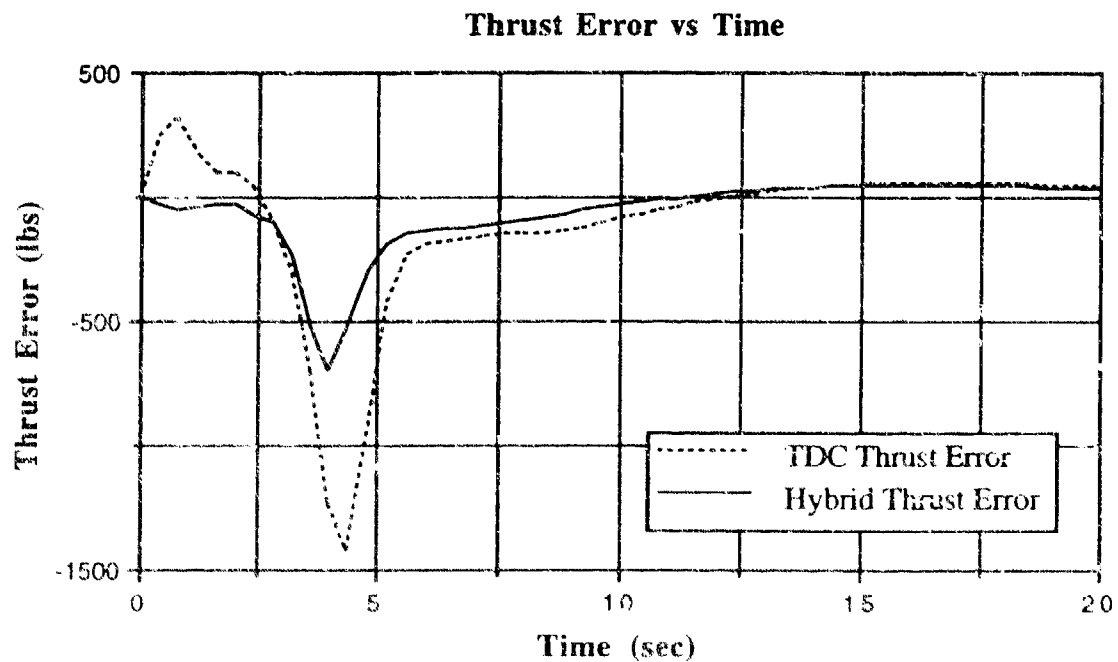


Figure 4.23(b) Error in thrust between reference trajectory and TDC or hybrid controlled aircraft.

ATTACHMENT 2

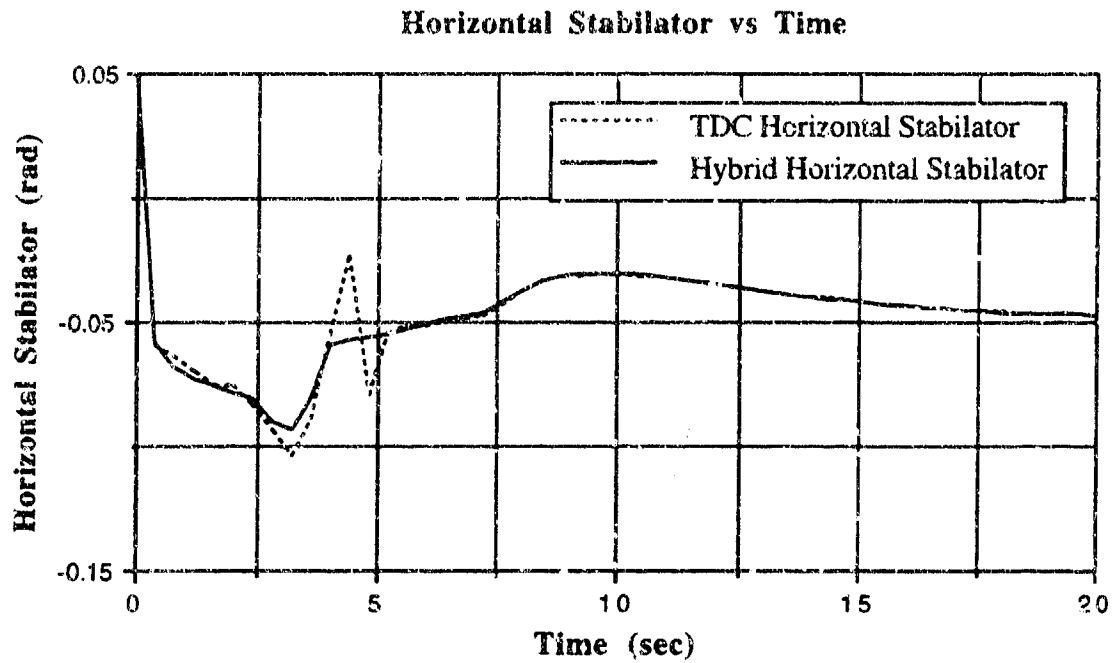


Figure 4.24 Horizontal stabilator deflection for the TDC and hybrid controlled aircraft.

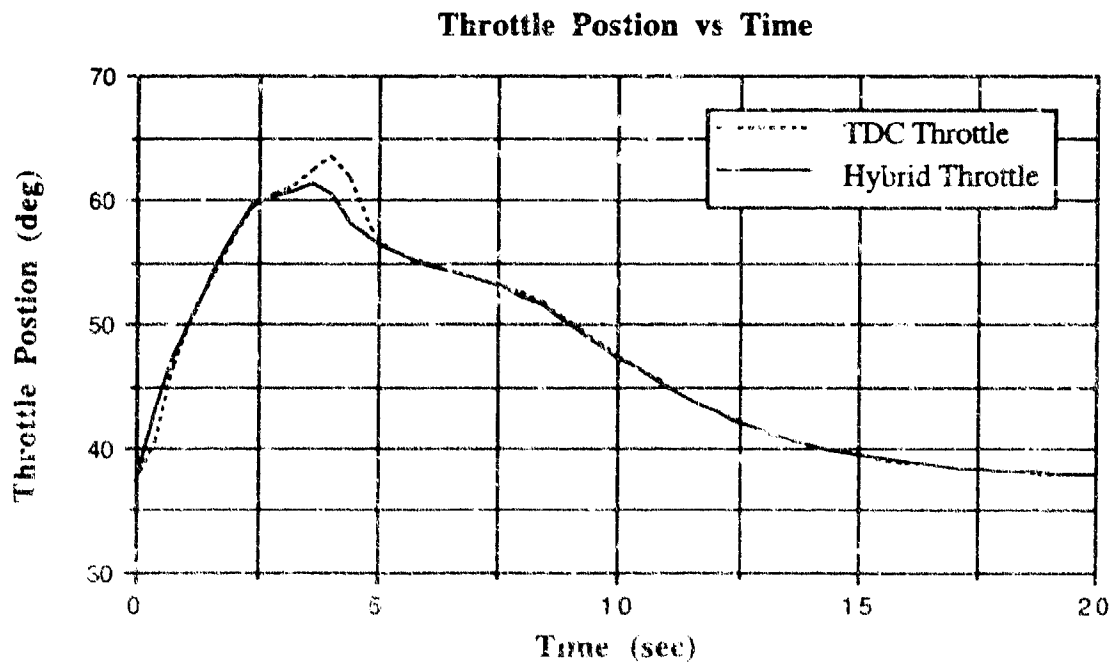


Figure 4.25 Throttle position for the TDC and hybrid controlled aircraft.

ATTACHMENT 2

Nonetheless, experiment 1 demonstrates that the hybrid controller is able to improve the performance of the aircraft over the purely adaptive TDC controller. This improved performance is realized by exploiting the learned functional mapping of the previously unknown model dynamics to remove the delay associated with the adaptive component and reduce the model uncertainty to arrive at a superior nonlinear control law. The next experiment illustrates the ability to generalize the synthesized mapping to a larger input space generated by using a more demanding commanded altitude rate.

4.2.6 High Performance Aircraft/Experiment 2

The objective of experiment 2 is to demonstrate the local generalization abilities of the learned functional mapping to areas of the input space that have not explicitly been trained. By increasing the rate limit on the randomly generated altitude command to 100 feet per second, the region of the input space for which controls must be computed is effectively increased. Moreover, the reference trajectory is more demanding in the sense that larger controls (resulting in larger angles and angular rates) are required to follow this trajectory.

Beyond the increased altitude rate limit, the setup of experiment 2 is identical to experiment 1 in terms of the learning system, initialization, and control calculation rate. Figures 4.26 through 4.31 contain the state trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft for a commanded 500 foot climb (at a 100 feet per second rate) and 10 feet per second increase in velocity using the *previously* trained network in experiment 1. The horizontal stabilator and throttle position applied to the aircraft for both the TDC and hybrid responses are shown in Figures 4.32 and 4.33.

ATTACHMENT 2

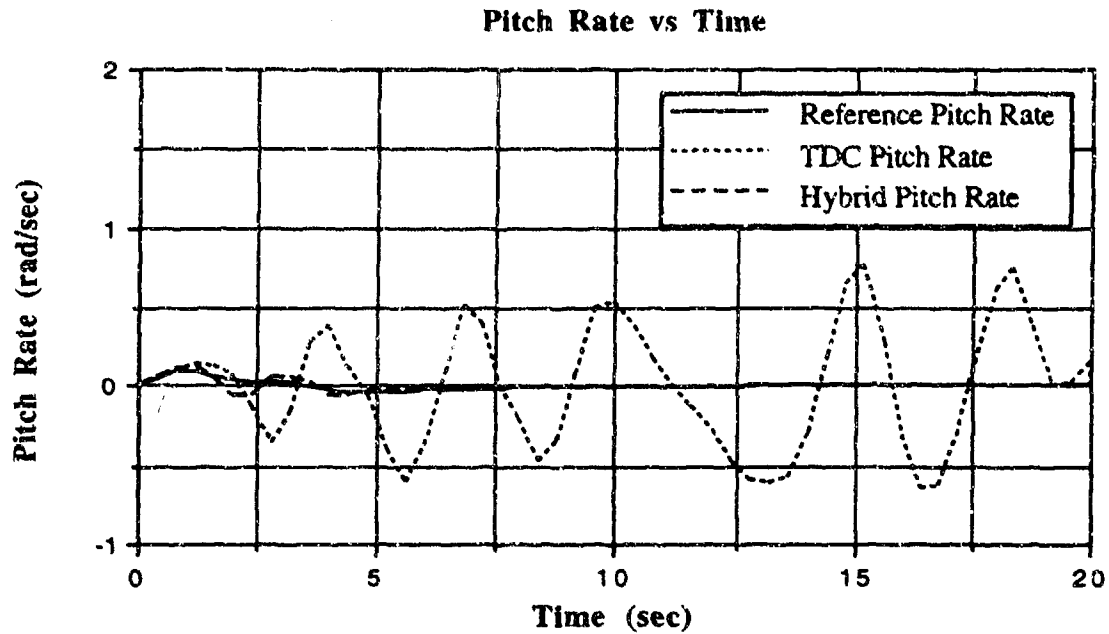


Figure 4.26 Pitch rate trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

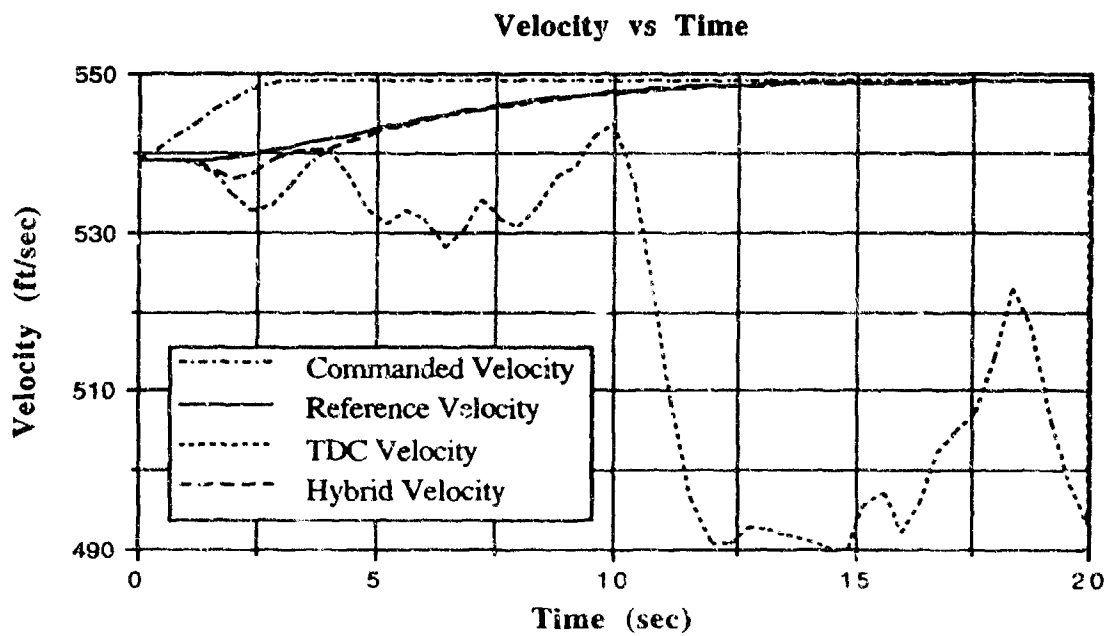


Figure 4.27 Velocity trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

ATTACHMENT 2

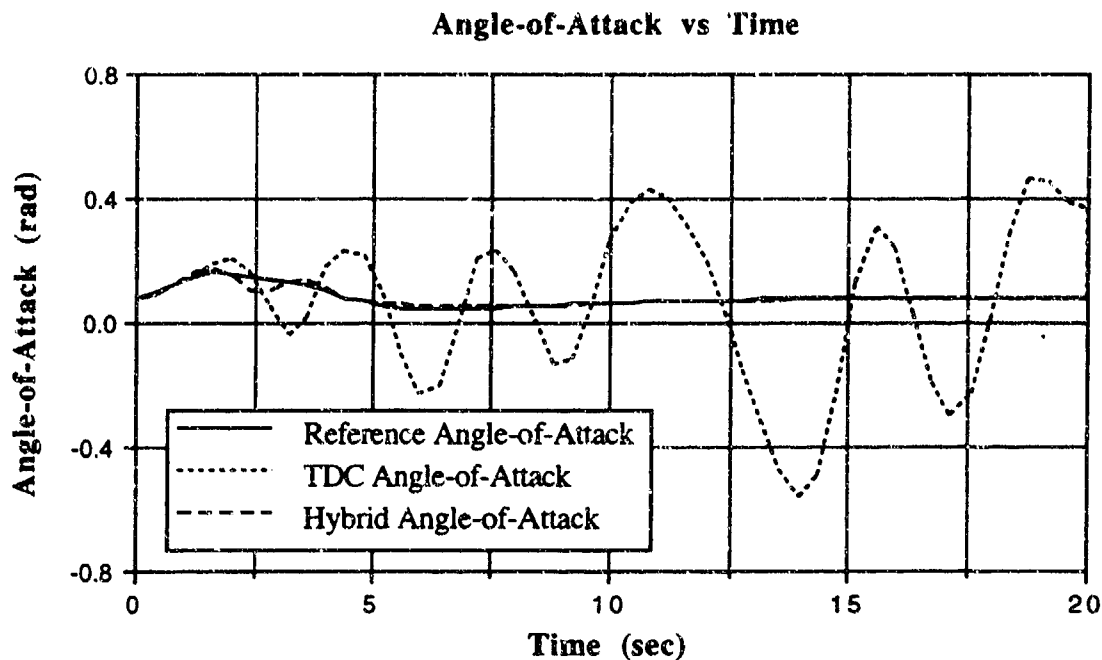


Figure 4.28 Angle-of-attack trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1

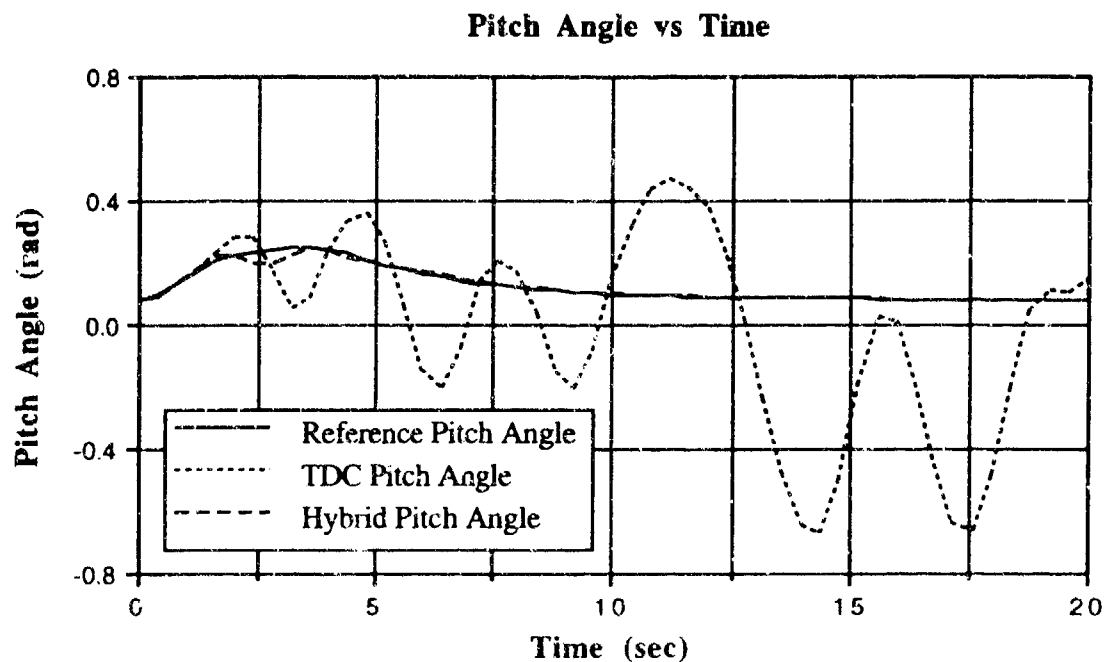


Figure 4.29 Pitch angle trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

ATTACHMENT 2

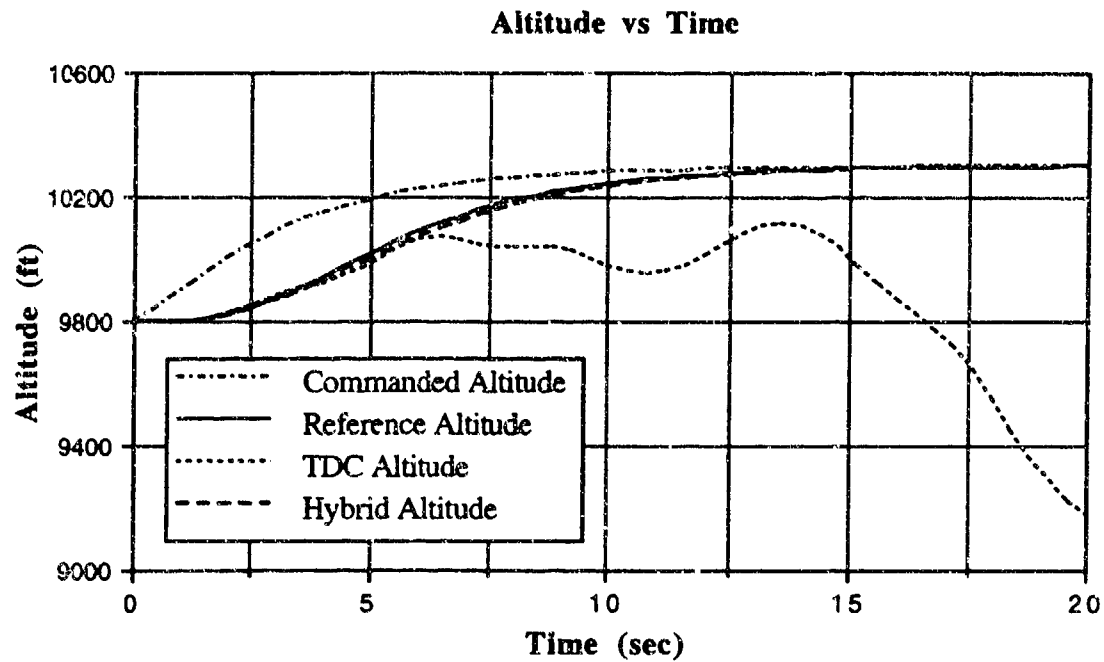


Figure 4.30 Altitude trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

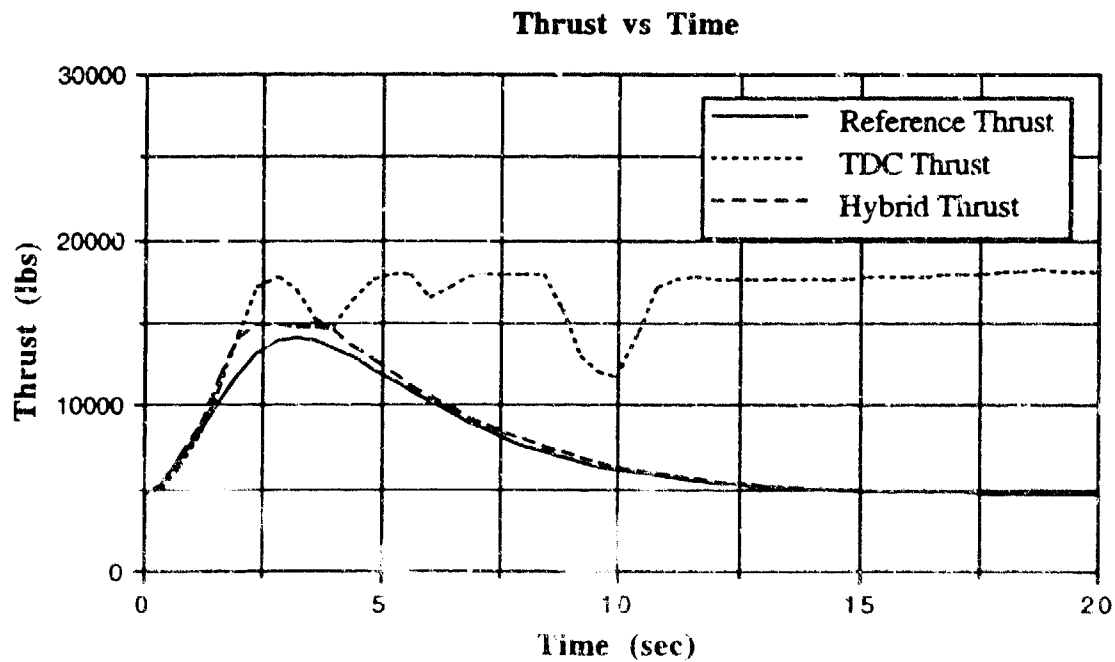


Figure 4.31 Thrust trajectories for the reference model, TDC controlled aircraft, and hybrid controlled aircraft using the network learned in experiment 1.

ATTACHMENT 2

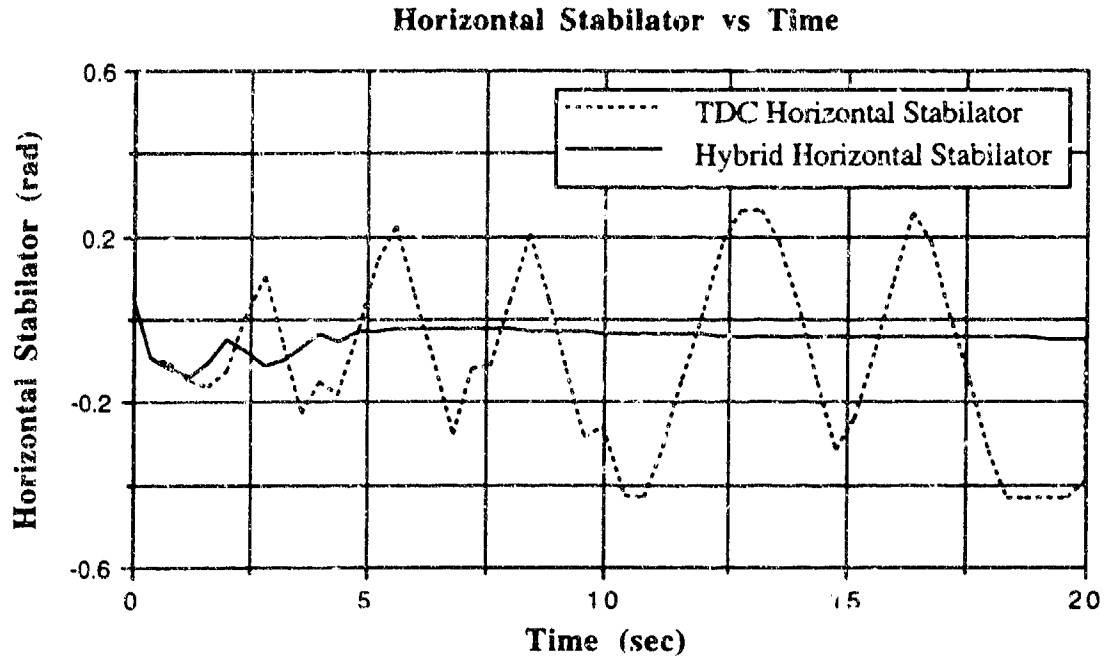


Figure 4.32 Horizontal stabilator deflection for the TDC and hybrid controlled aircraft.

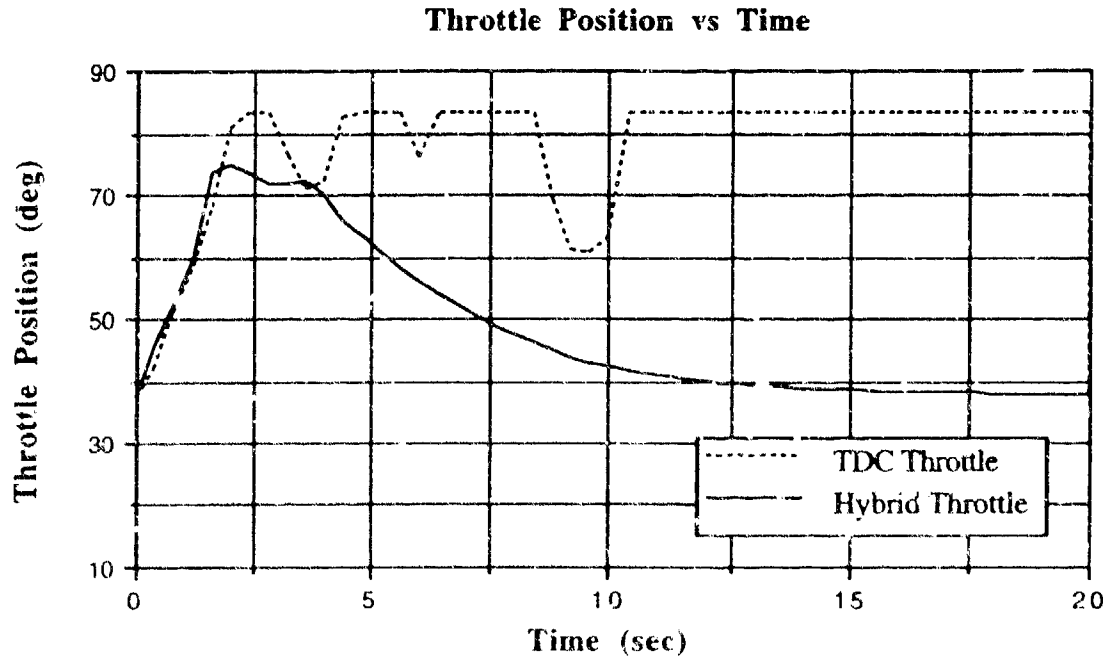


Figure 4.33 Throttle position for the TDC and hybrid controlled aircraft.

ATTACHMENT 2

As illustrated by the state trajectories, the TDC control law is unable to provide the control necessary to reach the commanded states, whereas the hybrid controlled aircraft generally follows the reference trajectory. Moreover, the banging of the horizontal stabilator and throttle against their limits for the TDC controller illustrates a desperate attempt to regain the desired state trajectory. This failure of TDC demonstrates the consequences of not using experientially gained knowledge to remove the delay in the estimate in the unknown dynamics and an inability to accommodate model uncertainty (e.g., improve the *a priori* estimate of the control weighting matrix).

Experiment 2 also demonstrates the ability of the learning system to generalize to nearby regions of the input space for which it has not explicitly received training samples. This feature is especially important due to that fact that the hybrid control law uses a passive learning system. Under passive learning, the learning system does not guide the vehicle in an active search of the input space. Instead, the learning system is opportunistic in the sense that it learns for a given region of the input space presented by the adaptive controller for the state trajectories that have been flown. As a result, areas of the input space in which TDC is unable to traverse can not initially receive training information. However, due to generalization, the hybrid controller is able to stabilize and control the aircraft in areas the purely adaptive control law fails. Later excursions through these regions will provide additional inputs for the learning system to process. This, of course, suggests a conservative approach to flight testing / learning if the hybrid controller were to be employed. Since the hybrid controller is able to adequately control the aircraft given an inaccurate linear representation, less *a priori* design information is needed (i.e., fewer design point linearizations), effectively reducing design costs and automating the tuning process.

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 SUMMARY AND CONCLUSIONS

This thesis describes the development and application of a hybrid control system to the problem of flight control for a high performance aircraft. By combining an adaptive component based on the TDC approach with a learning system, an innovative new hybrid controller has been formed that allows each of these two mechanisms to focus on the control objective for which it is best suited. The adaptive component of the hybrid controller accommodates some of the unmodeled dynamics and provides estimates of any unmodeled state dependent dynamic behavior to the learning system. The connectionist learning system synthesizes a functional approximation of the state dependent dynamic behavior. Using this learned mapping, the hybrid control system is able to predict state dependent behavior, effectively removing the delay an adaptive controller experiences due to its reactive nature.

The impact of a controller that has the ability to anticipate vehicle behavior has been illustrated in terms of improved closed-loop aircraft performance. It has also been shown that by using derivative information from the learned mapping, model uncertainty could be reduced at each operating condition, essentially automating the tuning process normally associated with gain scheduled controllers. Due to its ability to reduce model uncertainty, the hybrid system adequately controls the aircraft even in situations where an inaccurate linear representation was used as the system model during the initial design of the control law. As a result, less *a priori* design information is needed (i.e., fewer design point linearizations), effectively reducing design costs.

This thesis has also demonstrated the ability of a spatially localized learning system to synthesize a nonlinear, multivariable mapping in a control environment. More specifically, it has been shown that a linear-Gaussian network is able to learn a functional approximation of the initially unknown dynamics, given state and control information, using an incremental learning approach.

5.2 RECOMMENDATIONS FOR FUTURE WORK

The major constraint to the amount of improvement the hybrid control system could realize was not a function of the unknown dynamics or the ability of the learning system to synthesize this mapping, but the requirement that *all* the states follow their given reference trajectory. Since aircraft have more states than controls, this requirement is unrealistic from the control standpoint. Moreover, in many cases, only a few of the states are of direct importance. Further research following (Anderson & Schmidt (1990)) should focus on reducing the number of controlled states to be the same as the number of control inputs. Using this approach, the pseudo-inverse required in the derivation of the hybrid control law would be replaced by a true inverse, essentially allowing perfect model following for the case where all of the initially unknown dynamics are learned and there is no state and control observation noise.

Another area for future work is the expansion of the hybrid control system to map the entire flight envelope, as compared to a small subset of trajectories. This research would require a much larger network than that used for the experiments in this thesis, due to the expected nonlinearities in Mach number as well as angle-of-attack. A thorough examination of the abilities of the hybrid control law trained over the entire flight envelope could further highlight the advantages of this learning enhanced controller over conventional techniques.

A future investigation into using different types of adaptive components (i.e., other

ATTACHMENT 2

than TDC) in the hybrid control law is recommended (Astrom & Wittenmark (1989), Slotine & Li (1991)). Moreover, future research should examine areas of automatic flight control other than autopilots (e.g., stability augmentation systems and control augmentations systems) where the hybrid control law offers potential improvements.

BIBLIOGRAPHY

- Alexander, J., Baird, L., Baker, W. & Farrell, J. (1991). "A Design & Simulation Tool for Connectionist Learning Control Systems: Application to Autonomous Underwater Vehicles" Draper Laboratory Report CSDL-P-3041, Cambridge, MA.
- Albus, J. (1975). "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, pp. 220-227.
- Anderson, M. & Schmidt, D. (1990). "Error Dynamics and Perfect Model Following with Application to Flight Control", *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, pp. 912-919.
- Astrom, K. & Wittenmark, B. (1989). *Adaptive Control*, Addison-Wesley.
- Athans, M. (1990). "Lecture Notes for Multivariable Control Systems I," School of Engineering, MIT, Cambridge, MA.
- Athans, M., *et al.* (1977). "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method — Part I: Equilibrium Flight," *IEEE Transactions on Automatic Control*, Vol. 22, pp. 768-780.
- Baird, L. (1991). "Learning and Adaptive Hybrid Systems for Nonlinear Control," M.S. Thesis, Northeastern University, Boston, MA.
- Baker, W. & Farrell, J. (1990). "Connectionist Learning Systems for Control," Draper Laboratory Report CSDL-P-2922, Cambridge, MA.
- Baker, W. & Farrell, J. (1991). "Learning Augmented Flight Control for High Performance Aircraft," Draper Laboratory Report CSDL-P-3080, Cambridge, MA.
- Baker, W. & Farrell, J. (1992). "An Introduction to Connectionist Learning Control Systems," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, White, D. & Sofge, D. (eds.) Von Nostrand Reinhold.
- Barto, A. (1989). "Connectionist Learning for Control: An Overview," COINS Technical Report 89-89 Department of Computer and Information Science, University of Massachusetts, Amherst.
- Barto, A., Sutton, R., & Anderson, C. (1983). "Neuronlike Adaptive Elements That Can Solve Difficult Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, No. 5.
- Brumbaugh, R. (1991). "An Aircraft Model for the AIAA Controls Design Challenge," *Proceedings, AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA.

ATTACHMENT 2

- Bryson, A. & Ho, Y. (1975). *Applied Optimal Control*, Hemisphere.
- Duke, E. (1992). "AIAA Control Design Challenge."
- Etkin, B. (1982). *Dynamics of Flight - Stability and Control*, John Wiley and Sons.
- Farrell, J. & Baker, W. (1991). "Learning Control Systems," *Intelligent and Autonomous Control Systems*, Antsaklis, P. & Passino, K., (eds.), Kluiver Academic.
- Funahashi, K. (1988). "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp. 183-192.
- Hornik, K., Stinchcombe, M., & White, H. (1989). "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366.
- Jacobs, R. (1988). "Increased Rates of Convergence Through Learning Rate Adaptation", *Neural Networks*, Vol. 1, No. 4, pp. 295-307.
- Jordan, M. (1988). "Sequential Dependencies and Systems with Excess Degrees of Freedom," COINS Technical Report 88-27, University of Massachusetts, Amherst.
- Klopf, H. (1988). "A Neuronal Model of Classical Conditioning," *Psychobiology*, Vol. 16, pp. 85-125.
- Kwakernak, H. & Sivan, R. (1972). *Linear Optimal Control Systems*, John Wiley and Sons.
- Lewis, F. & Stevens, B. (1992). *Aircraft Control and Simulation*, John Wiley and Sons.
- Livstone, M., Farrell, J., & Baker, W. (1992). "A Computationally Efficient Algorithm for Training Recurrent Connectionist Networks," *Proceedings, 1992 American Control Conference*.
- Melsa, P. (1989). "Neural Networks: A Conceptual Overview," Tellabs Research Center, Mishawaka, IN.
- Mendel, J. & McLaren, R. (1970). "Reinforcement Learning and Pattern Recognition Systems," *Adaptive, Learning, and Pattern Recognition Systems: Theory and Applications*. Academic Press, New York.
- Millington, P. (1991). "Associative Reinforcement Learning for Optimal Control," S.M. Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- MIL-F-8785C (1980). *U.S. Department of Defense Military Specification: Flying Qualities of Piloted Airplanes*.
- Minsky, L. & Papert, S. (1969). *Perceptrons*, MIT Press, Cambridge, MA.
- Poggio, T. & Girosi, F. (1990). "Networks for Approximation and Learning," *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1481-1497.

ATTACHMENT 2

- Parkinson, G. & Smith, J. (1963). "The Square Prism as an Aeroelastic Non-Linear Oscillator," *Quarterly Journal of Mechanics and Applied Mathematics*, Vol. 17, pp. 225-239.
- Rumelhart, D., Hinton, G., & Williams, R. (1986). "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognitions, Vol. 1: Foundations*, MIT Press, Cambridge.
- Rosenblatt, F. (1962). *Principles of Neurodynamics*, Spartan Books, Washington.
- Roskam, J. (1979). *Airplane Flight Dynamics and Automatic Flight Controls*, Roskam Aviation and Engineering Corp.
- Slotine, J. & Li, W. (1991). *Applied Nonlinear Control*, Prentice-Hall.
- Stein, G. (1980). "Adaptive Flight Control: A Pragmatic View," in *Applications of Adaptive Control*, Narendra, K. & Monopoli, R. (eds.), Academic Press.
- Stein, G., Hartmann, G., & Hendrick, R. (1977). "Adaptive Control Laws for F-8 Flight Test," *IEEE Transactions on Automatic Control*, Vol. 22, pp. 758-767.
- Werbos, P. (1989). "Neural Networks for Control and System Identification," *Proceedings, IEEE Decision and Control Conference*, Tampa, FL.
- Widrow, B. & Hoff, M. (1960). "Adaptive Switching Circuits," *IRE Convention Record*, New York.
- Widrow, B. & Smith, F. (1964). "Pattern-Recognizing Control Systems," *Computer and Information Sciences Proceedings*, Washington, D.C.
- Yucef-Toumi, K. & Osamu, I. (1990). "A Time Delay Controller for Systems with Unknown Dynamics," *ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 112.

Reprint of:

Atkins, S. (1993). *Incremental Synthesis of Optimal Control Laws Using Learning Algorithms*, CSDL Report T-1181, S.M. Thesis, Department of Aeronautics and Astronautics, M.I.T.

INCREMENTAL SYNTHESIS OF OPTIMAL CONTROL LAWS USING LEARNING ALGORITHMS

Stephen C. Atkins

Submitted to the Department of Aeronautics and Astronautics
at the Massachusetts Institute of Technology
on May 7, 1993 in partial fulfillment of the requirements for the
degree of Master of Science in Aeronautics and Astronautics

ABSTRACT

Learning systems represent an approach to optimal control law design for situations where initial model uncertainty precludes the use of robust, fixed control laws. This thesis analyzes a variety of techniques for the incremental synthesis of optimal control laws, where the descriptor *incremental* implies that an on-line implementation filters the information acquired through real-time interactions with the plant and the operating environment. A *direct/indirect* framework is proposed as a means of classifying approaches to learning optimal control laws. Within this framework, relationships among existing direct algorithms are examined, and a specific class of indirect control laws is developed.

Direct learning control implies that the feedback loop that motivates the learning process is closed around system performance. Reinforcement learning is a type of direct learning technique with origins in the prediction of animal learning phenomena that is largely restricted to discrete input and output spaces. Three algorithms that employ the concept of reinforcement learning are presented: the Associative Control Process, Q learning, and the Adaptive Heuristic Critic.

Indirect learning control denotes a class of incremental control law synthesis methods for which the learning loop is closed around the system model. The approach discussed in this thesis integrates information from a learned mapping of the initially unmodeled dynamics into finite horizon optimal control laws. Therefore, the derivation of the control law structure as well as the closed-loop performance remain largely external to the learning process. Selection of a method to approximate the nonlinear function that represents the initially unmodeled dynamics is a separate issue not explicitly addressed in this thesis.

Dynamic programming and differential dynamic programming are reviewed to illustrate how learning methods relate to these classical approaches to optimal control design.

The aeroelastic oscillator is a two state mass-spring-dashpot system excited by a nonlinear lift force. Several learning control algorithms are applied to the aeroelastic oscillator to either regulate the mass position about a commanded point or to track a position reference trajectory; the advantages and disadvantages of these algorithms are discussed.

| | |
|-----------------------|---|
| Thesis Supervisor: | Professor Wallace E. Vander Velde |
| Title: | Department of Aeronautics and Astronautics |
| Technical Supervisor: | Walter L. Baker |
| Title: | Senior Member of Technical Staff, C. S. Draper Laboratory |

Acknowledgment

This thesis was prepared at the Charles Stark Draper Laboratory, Inc. with support provided by the U. S. Air Force Wright Laboratory under Contract F33615-88-C-1740. Publication of this report does not constitute approval by C. S. Draper Laboratory or the sponsoring agency of the findings or conclusions contained herein. This thesis is published for the exchange and stimulation of ideas.

Table of Contents

| | | |
|------------------|---|------------|
| Chapter 1 | Introduction | 361 |
| 1.1 | Problem Statement | 361 |
| 1.2 | Thesis Overview | 362 |
| 1.3 | Concepts | 364 |
| 1.3.1 | Optimal Control | 364 |
| 1.3.2 | Fixed and Adjustable Control | 365 |
| 1.3.3 | Adaptive Control | 366 |
| 1.3.4 | Learning Control | 367 |
| 1.3.5 | Generalization and Locality in Learning | 367 |
| 1.3.6 | Supervised and Unsupervised Learning | 368 |
| 1.3.7 | Direct and Indirect Learning | 369 |
| 1.3.8 | Reinforcement Learning | 369 |
| 1.3.9 | BOXES | 371 |
| Chapter 2 | The Aeroelastic Oscillator | 372 |
| 2.1 | General Description | 372 |
| 2.2 | The Equations of Motion | 374 |
| 2.3 | The Open-loop Dynamics | 379 |

ATTACHMENT 3

Table of Contents

| | |
|---|------------|
| 2.4 Benchmark Controllers | 381 |
| 2.4.1 Linear Dynamics | 382 |
| 2.4.2 The Linear Quadratic Regulator | 383 |
| 2.4.3 Bang-bang Controller | 384 |
| Chapter 3 The Associative Control Process | 389 |
| 3.1 The Original Associative Control Process | 390 |
| 3.2 Extension of the ACP to the Infinite Horizon, Optimal Control Problem | 401 |
| 3.3 Motivation for the Single Layer Architecture of the ACP | 405 |
| 3.4 A Single Layer Formulation of the Associative Control Process | 409 |
| 3.5 Implementation | 412 |
| 3.6 Results | 415 |
| Chapter 4 Policy and Value Iteration | 418 |
| 4.1 Terminology | 418 |
| 4.1.1 Total Discounted Future Return | 418 |
| 4.1.2 The Markov Decision Process | 419 |
| 4.1.3 Value Function | 419 |
| 4.1.4 Action Values | 420 |
| 4.2 Policy Iteration | 421 |
| 4.3 Value Iteration | 422 |
| 4.4 Q Learning | 423 |
| 4.5 Implementation | 424 |
| 4.6 Results | 425 |

ATTACHMENT 3

| | |
|---|------------|
| 4.7 Continuous Q Learning | 436 |
| Chapter 5 Temporal Difference Methods | 439 |
| 5.1 TD(λ) Learning Procedures | 440 |
| 5.2 An Extension of TD(λ) | 442 |
| 5.3 A Comparison of Reinforcement Learning Algorithms | 443 |
| Chapter 6 Indirect Learning Optimal Control | 447 |
| 6.1 Single-Stage Quadratic Optimization | 448 |
| 6.1.1 Linear Compensation | 449 |
| 6.1.2 Learning Control | 450 |
| 6.1.3 Penalizing Control Rate | 452 |
| 6.2 Two-Stage Quadratic Optimization | 453 |
| 6.2.1 Linear Compensation | 454 |
| 6.2.2 Learning Control | 455 |
| 6.2.3 Multistage Quadratic Optimization | 457 |
| 6.3 Implementation and Results | 458 |
| 6.3.1 Reference Model | 458 |
| 6.3.2 Function Approximation | 459 |
| 6.3.3 Single-Stage Quadratic Optimization Results | 462 |
| Chapter 7 Summary | 470 |
| 7.1 Conclusions | 471 |
| 7.1.1 Direct/Indirect Framework | 472 |
| 7.1.2 A Comparison of Direct Learning Algorithms | 473 |

ATTACHMENT 3

Table of Contents

| | |
|--|-----|
| 7.1.3 Limitations of Two-layer ACP Architectures | 473 |
| 7.1.4 Discussion of Differential Dynamic Programming | 474 |
| 7.2 Recommendations for Future Research | 474 |
| Appendix A Differential Dynamic Programming | 476 |
| A.1 Classical Dynamic Programming | 477 |
| A.2 Differential Dynamic Programming | 478 |
| Appendix B An Analysis of the AEO Open-loop Dynamics | 486 |
| References | 491 |

List of Figures

| | |
|--|-----|
| 2.1 The aeroelastic oscillator | 373 |
| 2.2 The aeroelastic oscillator nonlinear coefficient of lift | 375 |
| 2.3 The total velocity vector v_e and the effective angle of attack α | 376 |
| 2.4 The aeroelastic oscillator open-loop dynamics. An outer stable limit cycle surrounds an unstable limit cycle that in this picture decays inward to an inner stable limit cycle | 380 |
| 2.5 The AEO state trajectory achieved by a magnitude limited LQR control law | 385 |
| 2.6 The LQR control history and the limited force which yields Figure 2.5 | 385 |
| 2.7 The AEO state trajectory achieved by a LQR solution which was derived from a model with error in the linear dynamics. $x_0 = \{-1.0, 0.5\}$ | 386 |
| 2.8 The control history for the LQR solution which was derived from a model of the AEO with error in the linear dynamics. | 386 |
| 2.9 The AEO state trajectory achieved by a bang-bang control law derived from the IQR solution | 387 |
| 2.10 The control history of a bang-bang controller derived from the LQR solution, which yields Figure 2.9 | 387 |
| 3.1 The ACP network architecture | 392 |
| 3.2 The output equation nonlinearity, (3.2) | 394 |
| 3.3 The lower bound on excitatory weights, (3.11a) | 399 |

ATTACHMENT 3

List of Figures

| | |
|--|-----|
| 3.4 The upper bound on inhibitory weights, (3.11b) | 399 |
| 3.5 The single layer ACP architecture | 410 |
| 3.6 A state transition and reinforcement accumulation cartoon | 414 |
| 3.7 A characteristic AEO state trajectory achieved by a reinforcement learning algorithm prior to learning | 416 |
| 3.8 The AEO state trajectory achieved by the modified two-layer ACP after learning | 417 |
| 3.9 The AEO state trajectory achieved by the single layer ACP after learning | 417 |
| 4.1a A Cartesian representation of the two-action optimal control policy | 426 |
| 4.1b A polar representation of the two-action optimal control policy | 427 |
| 4.2 <i>Experiment 1</i> : Expected discounted future return (Q value) for each state-action pair | 428 |
| 4.3 <i>Experiment 2</i> : Expected discounted future return (Q value) for each state-action pair | 429 |
| 4.4 <i>Experiment 1</i> : State trajectory, $\underline{x}_0 = \{-1.0, 0.5\}$ | 431 |
| 4.5 <i>Experiment 1</i> : Control history, $\underline{x}_0 = \{-1.0, 0.5\}$ | 431 |
| 4.6 <i>Experiment 1</i> : State trajectory, $\underline{x}_0 = \{1.0, 0.5\}$ | 432 |
| 4.7 <i>Experiment 1</i> : Control history, $\underline{x}_0 = \{1.0, 0.5\}$ | 432 |
| 4.8 <i>Experiment 2</i> : State trajectory, $\underline{x}_0 = \{-1.0, 0.5\}$ | 433 |
| 4.9 <i>Experiment 2</i> : Control history, $\underline{x}_0 = \{-1.0, 0.5\}$ | 433 |
| 4.10 <i>Experiment 2</i> : State trajectory, $\underline{x}_0 = \{1.0, 0.5\}$ | 434 |
| 4.11 <i>Experiment 2</i> : Control history, $\underline{x}_0 = \{1.0, 0.5\}$ | 434 |
| 4.12 A continuous Q function for an arbitrary state \underline{x} | 436 |

ATTACHMENT 3

| | |
|--|-----|
| 6.1 The initially unmodeled dynamics $g(x_2)$ as a function of velocity x_2 | 461 |
| 6.2 Position and velocity time histories for the reference model as well as the AEO controlled by the linear and learning control laws, for the command $r = 1$ and the initial condition $\underline{x}_0 = \{0, 0\}$ | 463 |
| 6.3 The state errors for the AEO controlled by the linear and learning control laws | 464 |
| 6.4 The network outputs that were used to compute u_k for the learning control law | 464 |
| 6.5 The control u_k and the constituent terms of the learning control law (6.14) | 466 |
| 6.6 The control u_k and the constituent terms of the linear control law (6.7) | 466 |
| 6.7 The estimated errors in the approximation of the initially unmodeled dynamics $f_k(x_k, u_{k-1})$ | 467 |
| 6.8 AEO regulation from $\underline{x}_0 = \{-1.0, 0.5\}$ with control saturation at ± 0.5 | 468 |
| 6.9 Control history associated with Figure 6.8 | 468 |
| B.1 The steady-state amplitudes of oscillation \bar{X}_{ss} versus the incident windspeed U | 488 |
| B.2 $\frac{dR}{dU}$ versus R for $U = 2766.5$ | 489 |

Chapter 1

Introduction

1.1 Problem Statement

The primary objective of this thesis is to incrementally synthesize a nonlinear optimal control law, through real time, closed-loop interactions between the dynamic system, its environment, and a learning system, when substantial initial model uncertainty exists. The dynamic system is assumed to be nonlinear, time-invariant, and of known state dimension, but otherwise only inaccurately described by an a priori model. The problem, therefore, requires either explicit or implicit system identification. No disturbances, noise, or other time varying dynamics exist. The optimal control law is assumed to extremize an evaluation of the state trajectory and the control sequence, for any initial condition.

Chapter 1 - Introduction

1.2 Thesis Overview

One objective of this thesis is to present an investigation of several approaches for incrementally synthesizing (on-line) an optimal control law. A second objective is to propose a *direct/indirect* framework, with which to distinguish learning algorithms. This framework subsumes concepts such as supervised/unsupervised learning and reinforcement learning, which are not directly related to control law synthesis. This thesis unifies a variety of concepts from control theory and behavioral science (where the learning process has been considered extensively) by presenting two different learning algorithms applied to the same control problem: the Associative Control Process (ACP) algorithm [14], which was initially developed to predict animal behavior, and Q learning [16], which derives from the mathematical theory of value iteration.

The aeroelastic oscillator (§2), a two-state physical system that exhibits interesting nonlinear dynamics, is used throughout the thesis to evaluate different control algorithms which incorporate learning. The algorithms that are explored in §3, §4, and §5 do not explicitly employ dynamic models of the system and, therefore, may be categorized as direct methods of learning an optimal control law. In contrast, §6 develops an indirect, model-based, approach to learning an optimal control law.

The Associative Control Process is a specific reinforcement learning algorithm applied to optimal control, and a description of the ACP in §3 introduces the concept of direct learning of an optimal control law. The ACP, which includes a prominent network architecture, originated in the studies of animal behavior. The Q learning algorithm, which derives from the mathematical theorems of policy

ATTACHMENT 3

1.2 Thesis Overview

iteration and value iteration, is a simple reinforcement learning rule independent of a network architecture and of biological origins. Interestingly, Klopf's ACP [14] may be reduced so that the resulting system accomplishes Watkins' Q learning algorithm [16]. Sutton's theory of the temporal difference methods [15], presented in §5, subsumes the ACP and Q learning algorithms by generalizing the reinforcement learning paradigm applied to optimal control.

Several control laws that are optimal with respect to various finite horizon cost functionals are derived in §6 to introduce the indirect approach to learning optimal controls. The structure of the control laws with and without learning augmentation appears for several cost functionals, to illustrate the manner in which learning may augment a fixed parameter control design.

Finally, dynamic programming (DP) and differential dynamic programming (DDP) are reviewed in Appendix A as classical, alternative methods for synthesizing optimal controls. DDP is not restricted to operations in a discrete input space and discrete output space. The DP and DDP algorithms are model-based and, therefore, learning may be introduced by explicitly improving the a priori model, resulting in an indirect learning optimal controller. However, neither DP nor DDP is easily implemented on-line. Additionally, DDP does not address the problem of synthesizing a control law over the full state space.

Chapter 1 - Introduction

1.3 Concepts

The primary job of an automatic controller is to manipulate the inputs of a dynamic system so that the system behavior satisfies the stability and performance specifications which constitute the control objective. The design of such a control law may involve numerous difficulties, including multivariable, nonlinear, and time varying dynamics, with many degrees of freedom. Further design challenges arise from the existence of model uncertainty, disturbances and noise, complex objective functions, operational constraints, and the possibility of component failure. An examination of the literature reveals that control design methodologies typically address a subset of these issues while making simplifying assumptions to satisfy the remainder - a norm to which this thesis conforms.

This section is intended to introduce the reader to some of the relevant issues by previewing concepts that appear throughout the thesis and are peculiar to learning systems and control law development. Additionally, this section motivates the importance of learning control research.

1.3.1 Optimal Control

This thesis examines methods for synthesizing optimal control laws, the objective of which is to extremize a scalar functional evaluation of the state trajectory and control history. The solution of an optimal control problem generally requires the solution of a constrained optimization problem; the calculus of variations and dynamic programming address this issue. However, an optimal control rule may be evaluated by these methods only if an accurate model of the dynamics is available.

ATTACHMENT 3

1.3 Concepts

In the absence of a complete and accurate a priori model, these approaches may be applied to a model that is derived through observed objective function evaluations and state transitions; this constitutes indirect learning control. Alternatively, in environments with substantial initial uncertainty, direct learning control can be considered to perform incremental dynamic programming without explicitly estimating a system model [1].

1.3.2 Fixed and Adjustable Control

Most control laws may be classified into one of two broad categories: fixed or adjustable. The constant parameters of fixed control designs are selected using an a priori model of the plant dynamics. As a result, stability robustness to modeling uncertainty is potentially traded against performance; the attainable performance of the closed-loop system is limited by the accuracy of the a priori description of the equations of motion and statistical descriptions of noise and disturbances. Adjustable control laws incorporate real-time data to reduce, either explicitly or implicitly, model uncertainty, with the intention of improving the closed-loop response.

An adjustable control design becomes necessary in environments where the controller must operate in uncertain conditions or when a fixed parameter control law that performs sufficiently well cannot be designed from the limited a priori information. The two main classes of adjustable control are adaptation and learning; both reduce the level of uncertainty by filtering empirical data that is gained experientially [2,3].

Chapter 1 - Introduction

1.3.3 Adaptive Control

Noise and disturbances, which are present in all real systems, represent the unpredictable, time dependent features of the dynamics. Nonlinearities and coupled dynamics, which are predictable, spatial functions, constitute the remaining model errors.¹ Adaptive control techniques react to dynamics that appear to be time varying, while learning controllers progressively acquire spatially dependent knowledge about unmodeled dynamics. This fundamental difference in focus allows learning systems to avoid several deficiencies exhibited by adaptive algorithms in accommodating model errors. Whenever the plant operating condition changes, a new region of the nonlinear dynamics may be encountered. A memoryless adaptive control method must reactively adjust the control law parameters after observing the system behavior for the current condition, even if that operating condition has been previously experienced. The transient effects of frequently adapting control parameters may degrade closed-loop performance. A learning system, which utilizes memory to recall the appropriate control parameters as a function of the operating condition or state of the system, may be characterized as predictive rather than reactive.

Adaptive control exists in two flavors: direct and indirect. Indirect adaptive control methods calculate control actions from an explicit model of the system, which is enhanced with respect to the a priori description through a system identification procedure. Direct adaptive control methods modify the control system parameters without explicitly developing improvements in the initial system model.

¹ The term *spatial* implies a function that does not explicitly depend on time.

1.3 Concepts

While direct adaptive techniques to perform regulation and tracking are well established, adaptive optimal controllers are primarily indirect.

1.3.4 Learning Control

A learning control system is characterized by the automatic synthesis of a functional mapping through the filtering of information acquired during previous real-time interactions with the plant and operating environment [2]. With the availability of additional experience, the mapping of appropriate control actions as a function of state or the mapping of unmodeled dynamics as a function of state and control, is incrementally improved. A learning system, which is implemented using a general function approximation scheme, may either augment traditional fixed or adaptive control designs, or may operate independently.

1.3.5 Generalization and Locality in Learning

Generalization in a parameterized, continuous mapping implies that each adjustable parameter influences the mapping over a region of non-zero measure [4]. The effect of generalization in function synthesis is to provide automatic interpolation between training data. If the plant dynamics are continuous functions of time and state, then the control law will also be continuous. Therefore, the validity of generalization follows directly from the continuity of the dynamics and the desired control law [2].

The concept of locality of learning is related to generalization, but differs in scope. Locality of learning implies that a change in any single adjustable parameter will only alter the mapped function over a localized region of the input space. For

Chapter 1 - Introduction

non-localized learning, extensive training in a restricted region of the input space, which might occur when a system is regulated about a trim condition, can corrupt the previously acquired mapping for other regions. Therefore, on-line learning for which training samples may be concentrated in a specific region of the input space, requires the locality attribute [2,3,4].

1.3.6 Supervised and Unsupervised Learning

Learning procedures may be distinguished as supervised or unsupervised according to the type of instructional information provided by the environment. A supervised learning controller requires both a teacher that provides the desired system response and the cost functional which depends on the system output error [5]. Supervised learning control systems often form the error signal by comparing measured system characteristics with predictions generated by an internal model. The supervised learning process evaluates how each adjustable parameter, within the internal model, influences the error signal.

The class of unsupervised control designs learns through a scalar evaluative feedback signal, such as the measure generated by a cost function, that is less informative than the gradient vector of the cost with respect to each adjustable parameter. This type of learning is also referred to as learning with a critic. The scalar evaluation which accrues from performing a i action in a state does not indicate the best to perform any other action in that state. Therefore, even in the case of only two possible actions, an evaluative learning signal contains significantly less information than the feedback required by a supervised learning algorithm [6].

1.3 Concepts

1.3.7 Direct and Indirect Learning

The classifiers *direct* and *indirect* learning are borrowed from the concept of direct versus indirect adaptive control. Direct learning control implies the feedback loop that motivates the learning process is closed around system performance. Indirect learning control denotes that the learning loop is closed around the system model. Whereas in §3 – §5 the learning process is closed around system performance, in §6, the learning loop is closed around system model improvement, leaving the control law derivation and resulting system performance “open-loop.”

Direct learning approaches to optimal control law synthesis, which employ reinforcement learning techniques, are not readily applicable to the reference model tracking problem. The adjustable parameters in a reinforcement learning method encode an evaluation of the cost to complete the objective. In a tracking environment, the command objective changes and future values may not be known. Therefore, the cost to complete the objective changes and the application of methods from §3 – §5 is restricted to regulation.

Indirect learning approaches to optimal control primarily employ supervised learning algorithms. In contrast, direct learning methods for optimal control law synthesis principally employ unsupervised learning algorithms.

1.3.8 Reinforcement Learning

Originally conceived in the study of animal learning phenomena, reinforcement learning is a type of unsupervised learning that responds to a performance measure feedback signal referred to as the reinforcement, which may represent a re-

ATTACHMENT 3

Chapter 1 - Introduction

was for a cost. At each discrete time step, the controller observes the current state, selects and policies an action, observes the subsequent state, and receives reinforcement. The control objective is to maximize the expected sum of discounted future reinforcements. The probability of choosing an action that yields a large discounted future reinforcement should be increased; actions that lead to small discounted future reinforcement should be selected less frequently [1]. Reinforcement learning methods often acquire successful action sequences by constructing two complimentary functions: a *policy* function maps the states into appropriate control actions and an *evaluation* function maps the states into expectations of the discounted future reinforcement.

The study of connectionist learning methods has evolved from research in connectionist models to theories founded in the established disciplines of function approximation and optimization, reinforcement learning has been demonstrated to be a suitable technique for solving some stable, nonlinear, optimal control problems [2-7].

Reinforcement learning addresses the credit assignment problem, which refers to the necessity of determining which actions in a sequence are "responsible" for an increase or decrease in reinforcement. The problem is most severe in environments where evaluation feedback occurs infrequently. Additionally, the reinforcement learning process highlights the compromise between passive and active learning. Passive learning recognizes opportunities to extract any information that becomes available during the operation of the controlled system. In contrast, a control system using an active learning scheme actively seeks to gain information in regions where insufficient learning has occurred [8]. For on-line applications, that each action has

1.3 Concepts

an information collecting role implies a tradeoff between the expected gain of information, which is related to *future* performance, and the immediate reinforcement, which measures the *current* system performance [8].

1.3.9 BOXES

BOXES [8] is a simple implementation of a learning controller. The state space is discretized into disjoint regions, and the learning algorithm maintains an estimate of the appropriate control action for each region. Associated with any approach using a discrete input space is an exponential growth in the number of bins, as the state dimension or the number of quantization levels per state variable increases [3]. Therefore, quantization of the state space is seldom an efficient mapping technique and a learning algorithm that uses this strategy can generally only represent only a coarse approximation to a continuous control law. Although this lookup table technique facilitates some aspects of implementation, any parameterized function approximation scheme capable of representing continuous functions will be more efficient with respect to the necessary number of free parameters. Additionally, generalization is inherent to such continuous mappings [3]. A BOXES approach exhibits locality in learning, but does not generalize information across bin boundaries.

Chapter 2

The Aeroelastic Oscillator

2.1 General Description

A simple aeroelastic oscillator (AEO) may be modeled as a classical mass-spring-dashpot system with the addition of two external forces: an aerodynamic lift force and a control force (Figure 2.1). The mass, a rectangular block exposed to a steady wind, is constrained to translate in the direction normal to the vector of the incident wind and in the plane of the page in Figure 2.1. Specifications of the AEO plant are borrowed from Parkinson and Smith [9] as well as from Thompson and Stewart [10]. The low dimensionality of the dynamic state, which consists of the position $x(t)$ and the velocity of the mass, reduces the complexity of computer simulations and allows the system dynamics to be easily viewed in a two-dimensional phase plane. The AEO exhibits a combination of interesting nonlinear dynamics, generated by the nonlinear aerodynamic lift, and parameter

ATTACHMENT 3

2.1 The Equations of Motion

uncertainty that constitute a good context in which to study learning as a method of incrementally synthesizing an optimal control law. The control objective may be either regulating the state near the origin of the phase plane or tracking a reference trajectory.

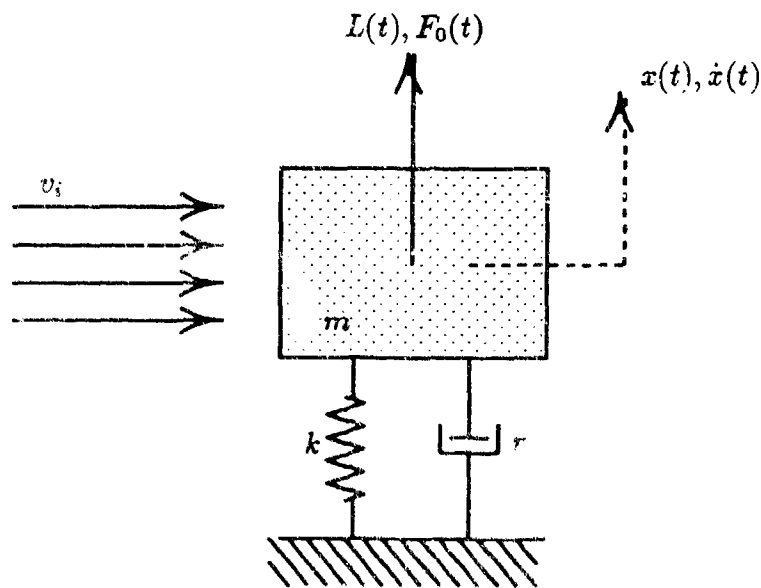


Figure 2.1. The aeroelastic oscillator.

2.2 The Equations of Motion

To investigate the AEO dynamics, the block is modeled as a point mass at which all forces act. The homogeneous equation of motion for the aeroelastic oscillator is a second order, linear, differential equation with constant coefficients. This equation accurately represents the physical system for zero incident windspeed, in

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

the range of block position and velocity for which the spring and dashpot respond linearly.

$$m \frac{d^2 x}{dt^2} + r \frac{dx}{dt} + kx = 0 \quad (2.1)$$

Table 2.1. Physical variable definitions.

| Physical Property | Symbol |
|---------------------|--------|
| Block Position | $x(t)$ |
| Block Mass | m |
| Damping Coefficient | r |
| Spring Coefficient | k |

For the undriven system, the block position may be described as a function of time by a weighted sum of exponentials whose powers are the roots of the characteristic equation.

$$x(t) = c_1 e^{s_1 t} + c_2 e^{s_2 t} \quad (2.2)$$

$$s_1, s_2 = \frac{-r \pm \sqrt{r^2 - 4mk}}{2m} = \frac{-r}{2m} \pm \frac{1}{2m} \sqrt{r^2 - 4mk} \quad (2.3)$$

$$k \geq 0 \quad \text{and} \quad r, m > 0 \Rightarrow \Re[s_1, s_2] \leq 0 \quad (2.4)$$

The condition that the dashpot coefficient is positive and the spring coefficient is non-negative, implies that the position and velocity transients will decay exponentially. This unforced system possesses a stable equilibrium at the origin of the phase plane.

ATTACHMENT 3

2.2 The Equations of Motion

The aerodynamic lift $L(t)$ and control force $F_0(t)$ constitute the driving component of the equation of motion. Including these forces, the equation of motion becomes a non-homogeneous, second-order, differential equation with constant coefficients.

$$m \frac{d^2 x}{dt^2} + r \frac{dx}{dt} + kx = L + F_0 \quad (2.5)$$

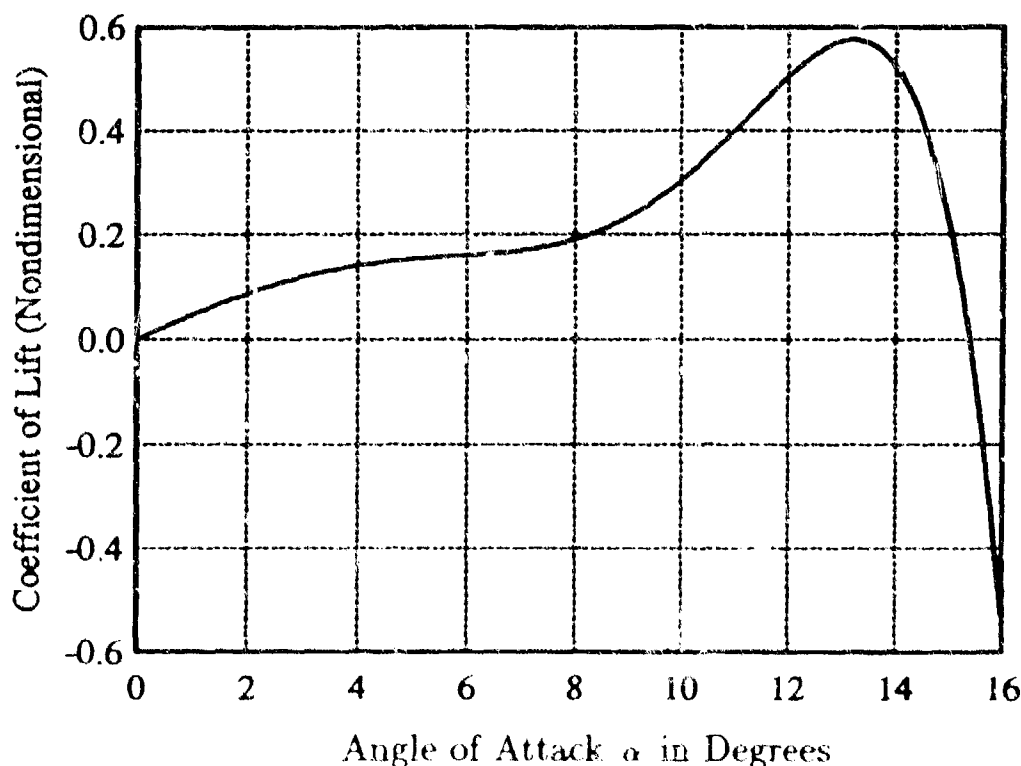


Figure 2.2. The aeroelastic oscillator nonlinear coefficient of lift.

The lift force is a nonlinear function of the effective angle of attack of the mass block with respect to the incident air flow. No current aerodynamic theory provides an analytic method for predicting the flow around an excited rectangular block. Therefore, the coefficient of lift is approximated, using empirical data, as a

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

seventh-order polynomial in the tangent of the effective angle of attack α (Figure 2.2) [9,10]. This approximation to the empirical data is valid for a range of angles of attack near zero degrees, $|\alpha| \leq 18^\circ$.

$$L = \frac{1}{2} \rho V^2 h l C_L \quad (2.6)$$

$$C_L = A_1 \left(\frac{\dot{x}}{V} \right) - A_3 \left(\frac{\dot{x}}{V} \right)^3 + A_5 \left(\frac{\dot{x}}{V} \right)^5 - A_7 \left(\frac{\dot{x}}{V} \right)^7 \quad (2.7)$$

$$\tan(\alpha) = \frac{\dot{x}}{V} \quad (2.8)$$

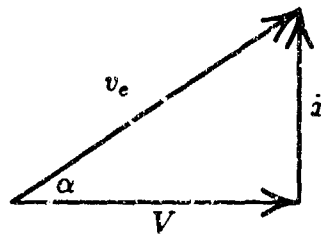


Figure 2.3. The total velocity vector v_e and the effective angle of attack α .

Table 2.2. Additional physical variable definitions.

| Physical Property | Symbol |
|-------------------------------------|--------|
| Density of Air | ρ |
| Velocity of Incident Wind | V |
| Area of Cross-section of Mass Block | $h l$ |
| Coefficient of Lift | C_L |

ATTACHMENT 3

2.2 The Equations of Motion

Following from the absence of even powers of $\frac{dx}{dt}$ in the polynomial (2.7), the coefficient of lift is an odd symmetric function of the angle of attack, which, given the geometry of the AEO, seems physically intuitive. The definition of the effective angle of attack is most apparent from the perspective that the AEO is moving through a stationary fluid. The total velocity v_e equals the sum of two orthogonal components: the velocity associated with the oscillator as a unit translating through the medium (i.e. the incident flow V), and the velocity \dot{x} associated with the mass block vibrating with respect to the fixed terminals of the spring and dashpot (Figure 2.3). This total velocity vector will form an effective angle of attack α with respect to the incident flow vector.

The dimensional equation of motion (2.5) can be nondimensionalized by dividing through by kh and applying the rules listed in Table 2.3. The resulting equation of motion may be written as (2.9) or equivalently (2.10).

$$\frac{d^2 X'}{d\tau^2} + 2\beta \frac{dX'}{d\tau} + X' = nA_1 U \frac{dX'}{d\tau} - \left(\frac{nA_3}{U} \right) \left(\frac{dX'}{d\tau} \right)^3 + \left(\frac{nA_5}{U^3} \right) \left(\frac{dX'}{d\tau} \right)^5 - \left(\frac{nA_7}{U^5} \right) \left(\frac{dX'}{d\tau} \right)^7 + F' \quad (2.9)$$

$$\frac{d^2 X'}{d\tau^2} + X' = nA_1 \left[\left(U - \left(\frac{2\beta}{nA_1} \right) \right) \frac{dX'}{d\tau} - \left(\frac{A_3}{A_1 U} \right) \left(\frac{dX'}{d\tau} \right)^3 + \left(\frac{A_5}{A_1 U^3} \right) \left(\frac{dX'}{d\tau} \right)^5 - \left(\frac{A_7}{A_1 U^5} \right) \left(\frac{dX'}{d\tau} \right)^7 \right] + F' \quad (2.10)$$

The coefficient of lift is parameterized by the following four empirically determined constants: $A_1 = 2.69$, $A_3 = 168$, $A_5 = 6270$, $A_7 = 59900$ [9,10]. The

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

other nondimensional system parameters were selected to provide interesting nonlinear dynamics: $n = 4.3 \cdot 10^{-4}$, $\beta = 1.0$, and $\frac{U}{U_c} = 1.6$. These parameters define $U_c = 1729.06$ and $U = 2766.5$, where the nondimensional critical windspeed U_c is defined in §2.3. The nondimensional time is expressed in radians.

Table 2.3. Required changes of variables.

| New Variables | Relationships |
|------------------------------|-------------------------------|
| Reduced displacement | $X' = \frac{x}{h}$ |
| Mass parameter | $n = \frac{\rho h^2 l}{2m}$ |
| Natural frequency | $\omega = \sqrt{\frac{k}{m}}$ |
| Reduced incident windspeed | $U = \frac{V}{\omega h}$ |
| Damping parameter | $\beta = \frac{r}{2m\omega}$ |
| Reduced time (radians) | $\tau = \omega t$ |
| Nondimensional Control Force | $F' = \frac{1}{kh} F_0$ |

The transformation from nondimensional parameters (n , β , and $\frac{U}{U_c}$) to dimensional parameters (ρ , h , l , m , V , r , and k) is not unique. Moreover, the nondimensional parameters that appear above will not transform to any physically realistic set of dimensional parameters. However, this set of nondimensional parameters creates fast dynamics which facilitates the analysis of learning techniques.

An additional change of variables scales the maximum amplitudes of the block's displacement and velocity to approximately unity in order of magnitude. The dynamics that are used throughout this thesis for experiments with the aeroe-

ATTACHMENT 3

2.3 The Open-loop Dynamics

lastic oscillator appear in (2.12).

$$X = \frac{X'}{1000} \quad F = \frac{F'}{1000} \quad (2.11)$$

$$\begin{aligned} \frac{d^2 X}{d\tau^2} + 2\beta \frac{dX}{d\tau} + X = \frac{1}{1000} \left[1000nA_1 U \frac{dX}{d\tau} - \left(\frac{nA_3}{U} \right) \left(1000 \frac{dX}{d\tau} \right)^3 \right. \\ \left. + \left(\frac{nA_5}{U^3} \right) \left(1000 \frac{dX}{d\tau} \right)^5 - \left(\frac{nA_7}{U^5} \right) \left(1000 \frac{dX}{d\tau} \right)^7 \right] + F \end{aligned} \quad (2.12)$$

Equation (2.12) may be further rewritten as a pair of first-order differential equations in a state space realization. Although in the dimensional form $\dot{x} = \frac{dx}{dt}$, in the nondimensional form, $\dot{X} = \frac{dX}{d\tau}$.

$$x_1 = X \quad x_2 = \frac{dX}{d\tau} \quad \dot{x}_1 = x_2 \quad \dot{x}_2 = \frac{d^2 X}{d\tau^2} \quad (2.13)$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & nA_1 U - 2\beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} F + \begin{bmatrix} 0 \\ f(x_2) \end{bmatrix} \quad (2.14a)$$

$$f(x_2) = \frac{1}{1000} \left[-\frac{nA_3}{U} (1000x_2)^3 + \frac{nA_5}{U^3} (1000x_2)^5 - \frac{nA_7}{U^5} (1000x_2)^7 \right] \quad (2.14b)$$

2.3 The Open-loop Dynamics

The reduced critical windspeed U_c , which depends on the nondimensional mass parameter, the damping parameter, and the first-order coefficient in the coefficient of lift polynomial, is the value of the incident windspeed at which the negative

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

linear aerodynamic damping exceeds the positive structural damping.¹

$$U_c = \frac{2\beta}{nA_1} \quad (2.15)$$

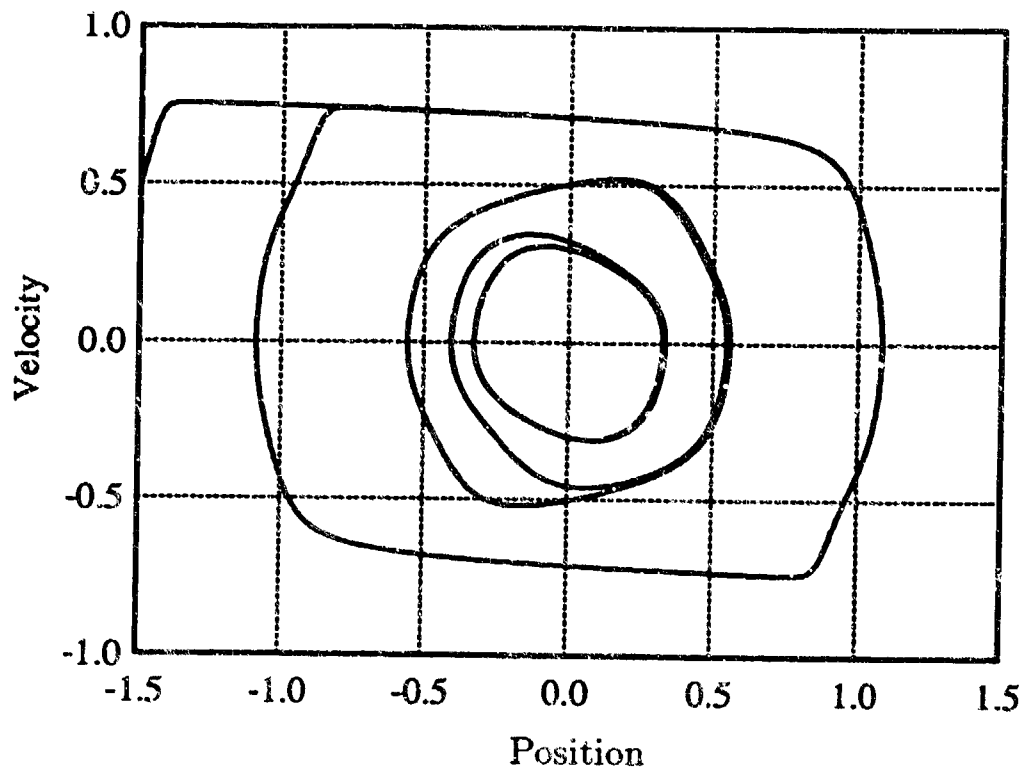


Figure 2.4. The aeroelastic oscillator open-loop dynamics. An outer stable limit cycle surrounds an unstable limit cycle that in this picture decays inward to an inner stable limit cycle.

The nature of the open-loop dynamics is strongly dependent on the ratio of the reduced incident windspeed to the reduced critical windspeed. At values of the incident windspeed below the critical value, the focus of the phase plane is stable

¹ The term *reduced* is synonymous with nondimensional.

2.4 Benchmark Controllers

and the state of the oscillator will return to the origin from any perturbed initial condition. For windspeeds greater than the critical value, the focus of the two dimensional state space is locally unstable; the system will oscillate, following a stable limit cycle clockwise around the phase plane. The aeroelastic oscillator is globally stable, in a bounded sense, for all U .² The existence of global stability is suggested by the coefficient of lift curve (Figure 2.2); the coefficient of lift curve predicts zero lift ($C_L = 0$) for $\alpha = \pm 15.3^\circ$ and a restoring lift force for larger $|\alpha|$. That the aeroelastic oscillator is globally open-loop stable eliminates the necessity for a feedback loop to provide nominal stability during learning experiments. For incident windspeeds greater than U_c , a limit cycle is generated at a stable Hopf bifurcation. In this simplest form of dynamic bifurcation, a stable focus bifurcates into an unstable focus surrounded by a stable limit cycle under the variation of a single independent parameter, U_c . For a range of incident wind velocity, two stable limit cycles, separated by an unstable limit cycle, characterize the dynamics (Figure 2.4). Figure 2.4 was produced by a 200Hz simulation in continuous time of the AEO equations of motion, using a fourth-order Runge-Kutta integration algorithm. An analysis of the open-loop dynamics appears in Appendix B.

2.4 Benchmark Controllers

A simulation of the AEO equations of motion in continuous time was implemented in the *NetSim* environment. *NetSim* is a general purpose simulation and

² Each state trajectory is a member of L_∞ (i.e. $\|\underline{x}(t)\|_\infty$ is finite) for all perturbations δ with bounded Euclidean norms, $\|\delta\|_2$.

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

design software package developed at the C. S. Draper Laboratory [11]. Ten *NetSim* cycles were completed for each nondimensional time unit while the equations of motion were integrated over twenty steps using a fourth-order Runge Kutta algorithm for each *NetSim* cycle.

Two simple control laws, based on a linearization of the AEO equations of motion, will serve as benchmarks for the learning controllers of §3, §4 and §5.

2.4.1 Linear Dynamics

From (2.14a), the linear dynamics about the origin may be expressed by (2.16) where A and B are given in (2.17).

$$\dot{\underline{x}}(\tau) = A\underline{x}(\tau) + \underline{B}u(\tau) \quad (2.16)$$

$$A = \begin{bmatrix} 0 & 1 \\ -1 & nA_1U - 2\beta \end{bmatrix} \quad \underline{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.17)$$

This linearization may be derived by defining a set of perturbation variables, $\underline{x}(\tau) = \underline{x}_0 + \delta\underline{x}(\tau)$ and $u(\tau) = u_0 + \delta u(\tau)$, which must satisfy the differential equations. Notice that $\delta\dot{\underline{x}}(\tau) = \dot{\underline{x}}(\tau)$. The expansion of $\delta\underline{x}(\tau)$ in a Taylor series about (\underline{x}_0, u_0) yields (2.18).

$$\begin{aligned} \delta\dot{\underline{x}}(\tau) &= \underline{f}[\underline{x}_0 + \delta\underline{x}(\tau), u_0 + \delta u(\tau)] \\ &= \underline{f}(\underline{x}_0, u_0) + \left. \frac{\partial \underline{f}}{\partial \underline{x}} \right|_{\underline{x}_0, u_0} \delta\underline{x}(\tau) + \left. \frac{\partial \underline{f}}{\partial u} \right|_{\underline{x}_0, u_0} \delta u(\tau) + \dots \end{aligned} \quad (2.18)$$

If the pair (\underline{x}_0, u_0) represents an equilibrium of the dynamics, then $\underline{f}(\underline{x}_0, u_0) = 0$ by definition. Equation (2.16) is achieved by discarding the nonlinear terms of

ATTACHMENT 3

2.4 Benchmark Controllers

(2.18) and applying (2.19), where A and B are the Jacobian matrices.

$$A = \left. \frac{\partial f}{\partial \underline{x}} \right|_{\underline{x}_0, u_0} \quad B = \left. \frac{\partial f}{\partial u} \right|_{\underline{x}_0, u_0} \quad (2.19)$$

2.4.2 The Linear Quadratic Regulator

The LQR solution minimizes a cost functional J that is an infinite time horizon integral of a quadratic expression in state and control. The system dynamics must be linear. The optimal control is given by (2.21)

$$J = \int_0^\infty [\underline{x}(\tau)^T \underline{x}(\tau) + u^2(\tau)] d\tau \quad (2.20)$$

$$u'(\tau) = -\underline{G}^T \underline{x}(\tau) \quad (2.21)$$

$$\underline{G} = \begin{bmatrix} 0.4142 \\ 3.0079 \end{bmatrix} \quad (2.22)$$

The actuators which apply the control force to the AEO are assumed to saturate at ± 0.5 nondimensional force units. Therefore, the control law tested in this section was written as

$$u(\tau) = f(-[0.4142 \quad 3.0079] \underline{x}(\tau)). \quad (2.23)$$

$$f(x) = \begin{cases} 0.5, & \text{if } x > 0.5 \\ -0.5, & \text{if } x < -0.5 \\ x, & \text{otherwise.} \end{cases} \quad (2.24)$$

The state trajectory which resulted from applying the control law (2.23) to the AEO, for the initial conditions $\{-1.0, 0.5\}$, appears in Figure 2.5. The controller

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

applied the maximum force until the state approached the origin, where the dynamics are nearly linear (Figure 2.6). Therefore, the presence of the nonlinearity in the dynamics did not strongly influence the performance of this control law.

If the linear dynamics were modeled perfectly (as above) and the magnitude of the control were not limited, the LQR solution would perform extremely well. Model uncertainty was introduced into the a priori model by designing the LQR controller assuming the open-loop poles were $0.2 \pm 1.8j$.

$$A' = \begin{bmatrix} 0 & 1 \\ -3.28 & 0.4 \end{bmatrix} \quad (2.26)$$

The LQR solution of (2.20) using A' is $\underline{G}^T = [0.1491, 1.6075]$. This control law applied to the AEO, when the magnitude of the applied force was limited at 0.5, produced the results shown in Figures 2.7 and 2.8. The closed-loop system was significantly under-damped.

2.4.3 Bang-bang Controller

The bang-bang controller was restricted to two control actions, a maximum positive force (0.5 nondimensional units) and a maximum negative force (-0.5); this limitation will also be imposed on the initial direct learning experiments. The control law is derived from the LQR solution and is non-optimal for the AEO system. In the half of the state space where the LQR solution specifies a positive force, the bang-bang control law (2.25) applies the maximum positive force. Similarly, in the half of the state space where the LQR solution specifies a negative force, the bang-bang control law applies the maximum negative force. The switching line which

ATTACHMENT 3

2.4 Benchmark Controllers

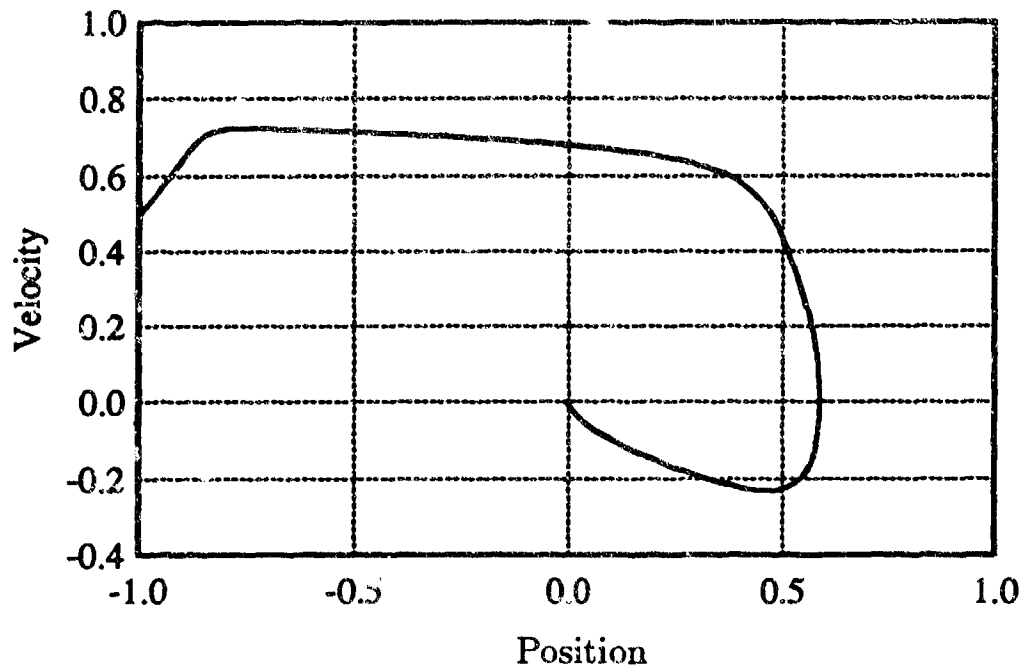


Figure 2.5. The AEO state trajectory achieved by a magnitude limited LQR control law.

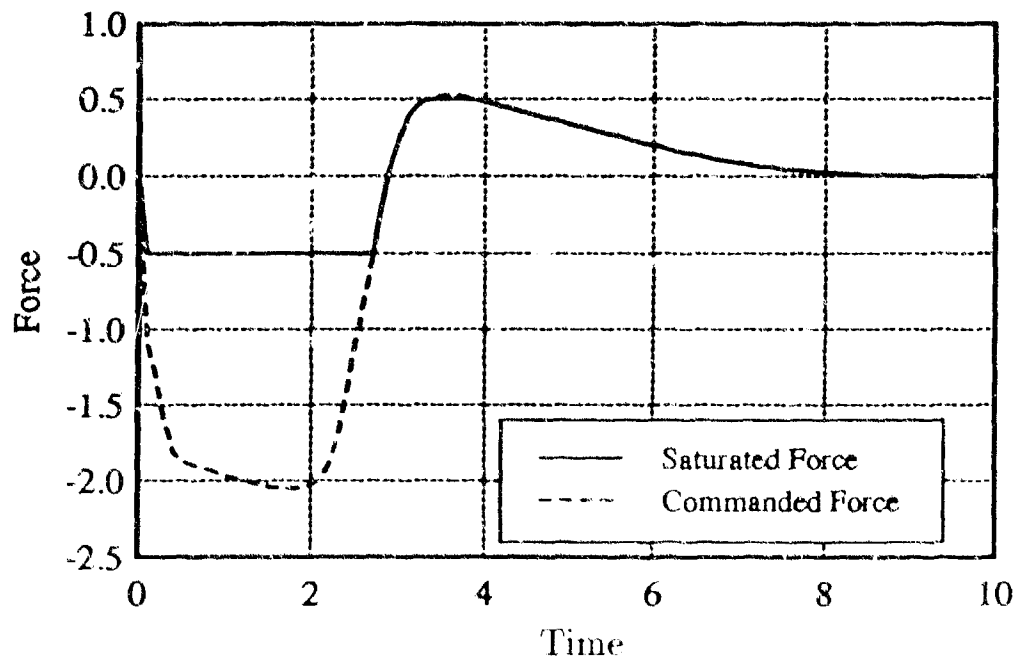


Figure 2.6. The LQR control history and the limited force which yields Figure 2.5.

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

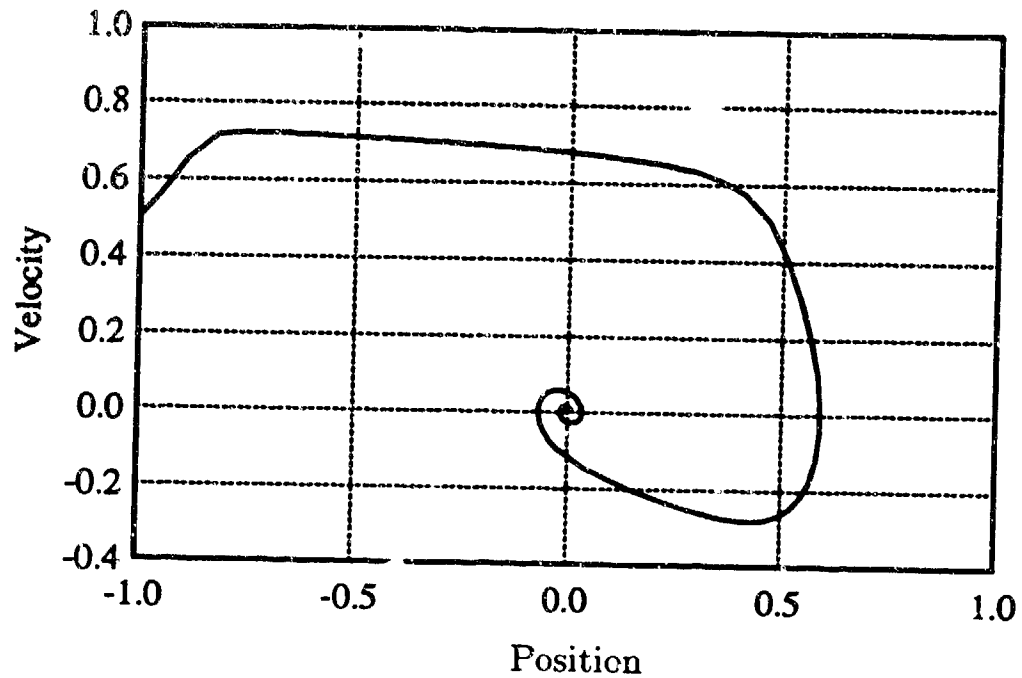


Figure 2.7. The AEO state trajectory achieved by a LQR solution which was derived from a model with error in the linear dynamics. $\mathbf{x}_0 = \{-1.0, 0.5\}$.

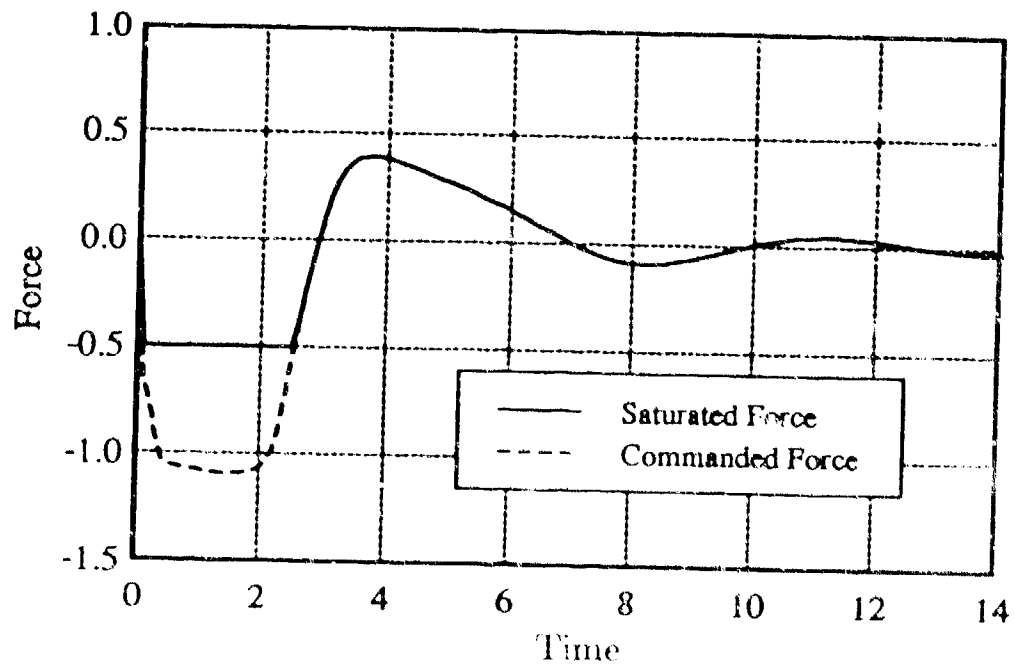


Figure 2.8. The control history for the LQR solution which was derived from a model of the AEO with error in the linear dynamics.

ATTACHMENT 3

2.4 Benchmark Controllers

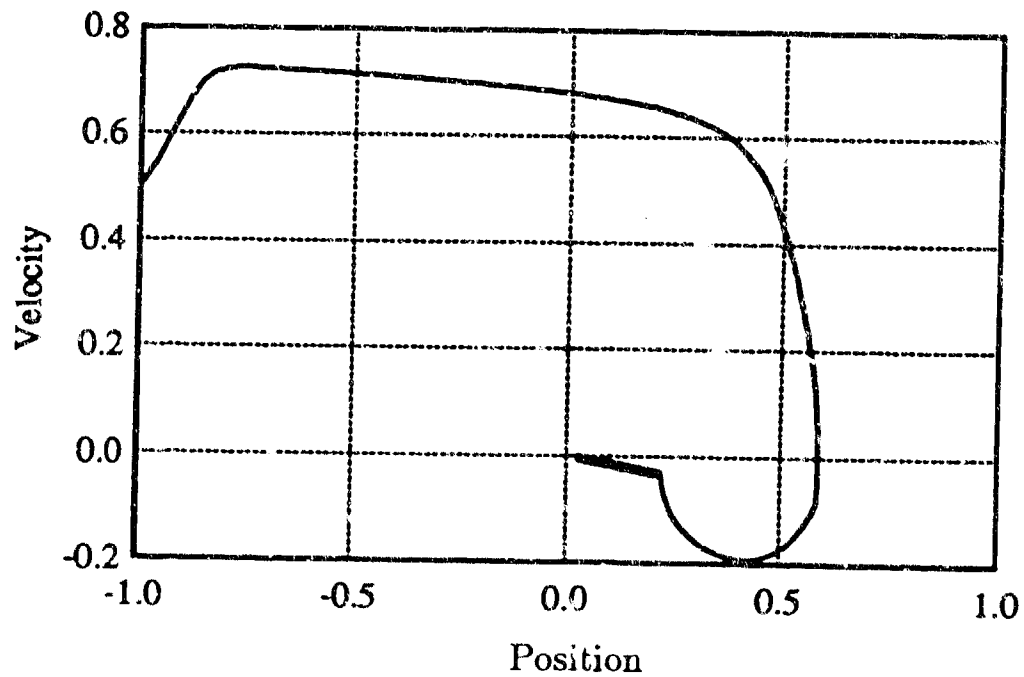


Figure 2.9. The AEO state trajectory achieved by a bang-bang control law derived from the LQR solution.

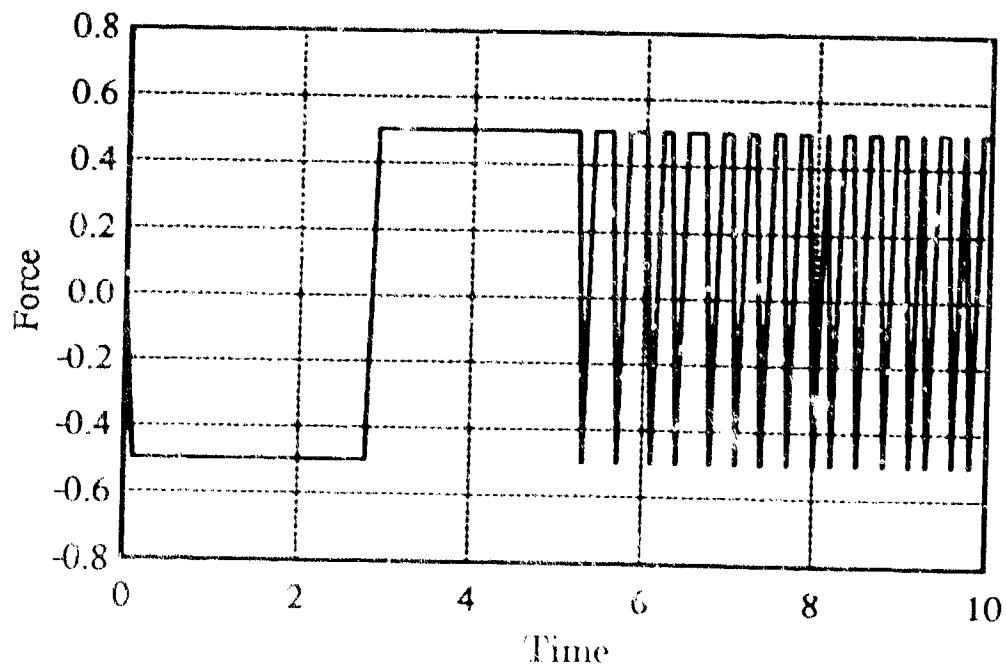


Figure 2.10. The control history of a bang-bang controller derived from the LQR solution, which yields Figure 2.9.

ATTACHMENT 3

Chapter 2 - The Aeroelastic Oscillator

divides the state space passes through the origin with slope -0.138 ; this is the line of zero force in the LQR solution.

$$u(\tau) = \begin{cases} 0.5, & \text{if } -\mathbf{G}^T \mathbf{x}(\tau) > 0 \\ -0.5, & \text{otherwise.} \end{cases} \quad (2.25)$$

The result of applying this bang-bang control law to the AEO with initial conditions $\{-1.0, 0.5\}$ appears in Figure 2.7. The trajectory initially traces the trajectory in Figure 2.5 because the LQR solution was saturated at -0.5 . However, the trajectory slowly converges toward the origin along the line which divides the positive and negative control regions, while rapidly alternating between exerting the maximum positive force and maximum negative force (Figure 2.8). Generally, this would represent unacceptable performance. The bang-bang control law represents a two-action, linear control policy and will serve as a non-optimal benchmark with which to compare the direct learning control laws. The optimal two-action control law cannot be written from only a brief inspection of the nonlinear dynamics.

Chapter 3

The Associative Control Process

The Associative Control Process (ACP) network [12,14] models certain fundamental aspects of the animal nervous system, accounting for numerous classical and instrumental conditioning phenomena.¹ The original ACP network was intended to model limbic system, hypothalamic, and sensorimotor function as well as to provide a general framework within which to relate animal learning psychology and control theory. Through real-time, closed-loop, goal seeking interactions between the learning system and the environment, the ACP algorithm can achieve solutions to spatial and temporal credit assignment problems. This capability suggests that the ACP algorithm, which accomplishes reinforcement or self-supervised learning, may offer solutions to difficult optimal control problems.

¹ Animal learning phenomena are investigated through two classes of laboratory conditioning procedures. *Classical conditioning* is an open-loop process in which the experience of the animal is independent of the behavior of the animal. The experience of the animal in closed-loop *instrumental conditioning* or *operant conditioning* experiments is contingent on the animal's behavior [12].

Chapter 3 - The Associative Control Process

This chapter constitutes a thorough description of the ACP network, viewed from the perspective of applying the architecture and process as a controller for dynamic systems.² A detailed description of the architecture and functionality of the original ACP network (§3.1) serves as a foundation from which to describe two levels of modification, intended to improve the applicability of the Associative Control Process to optimal control problems. Initial modifications to the original ACP specifications retain a two-layer network structure (§3.2); several difficulties in this *modified* ACP motivate the development of a single layer architecture. A single layer formulation of the ACP network abandons the biologically motivated network structure while, preserving the mathematical basis of the modified ACP (§3.4). This minimal representation of an Associative Control Process performs an incremental value-iteration procedure similar to Q learning and is guaranteed to converge to the optimal policy in the infinite horizon optimal control problem under certain conditions [13]. This chapter concludes with a summary of the application of the modified and single layer ACP methods to the regulation of the aeroelastic oscillator (§3.5 and §3.6).

3.1 The Original Associative Control Process

The definition of the original ACP is derived from Klopf [12], Klopf, Morgan, and Weaver [14], as well as Baird and Klopf [13]. Although originally introduced in the literature as a model to predict a variety of animal learning results from classical

² This context is in contrast to the perspective that an ACP network models aspects of biological systems.

3.1 The Original ACP

and instrumental conditioning experiments, a recast version of the ACP network has been shown to be capable of learning to optimally control any non-absorbing, finite-state, finite-action, discrete time Markov decision process [13]. Although the original form of the ACP may be incompatible with infinite time horizon optimal control problems, as an introduction to the ACP derivatives, the original ACP appears here with an accent toward applying the learning system to the optimal control of dynamic systems. Where appropriate, analogies to animal learning results motivate the presence of those features of the original ACP architecture which emanate from a biological origin. Although the output and learning equations are central in formalizing the ACP system, to eliminate ambiguities concerning the interconnection and functionality of network elements, substantial textual description of rules is required.

The ACP network consists of five distinct elements: *acquired drive sensors*, *motor centers*, *reinforcement centers*, *primary drive sensors*, and *effectors* (Figure 3.1). In the classical conditioning nomenclature, the acquired drive sensors represent the conditioned stimuli; in the context of a control problem, the acquired drive sensors encode the sensor measurements and will be used to identify the discrete dynamic state. The ACP requires an interface with the environment that contains a finite set of states. Therefore, for the application of the ACP to a control problem, the state space of a dynamic system is quantized into a set of m disjoint, non-uniform bins which fill the entire state space.³ The ACP learning system operates in discrete time. At any stage in discrete time, the state of the dynamic system

³ A sufficient condition is for the bins to fill the entire operational envelope, i.e. the region of the state space that the state may enter.

ATTACHMENT 3

Chapter 3 - The Associative Control Process

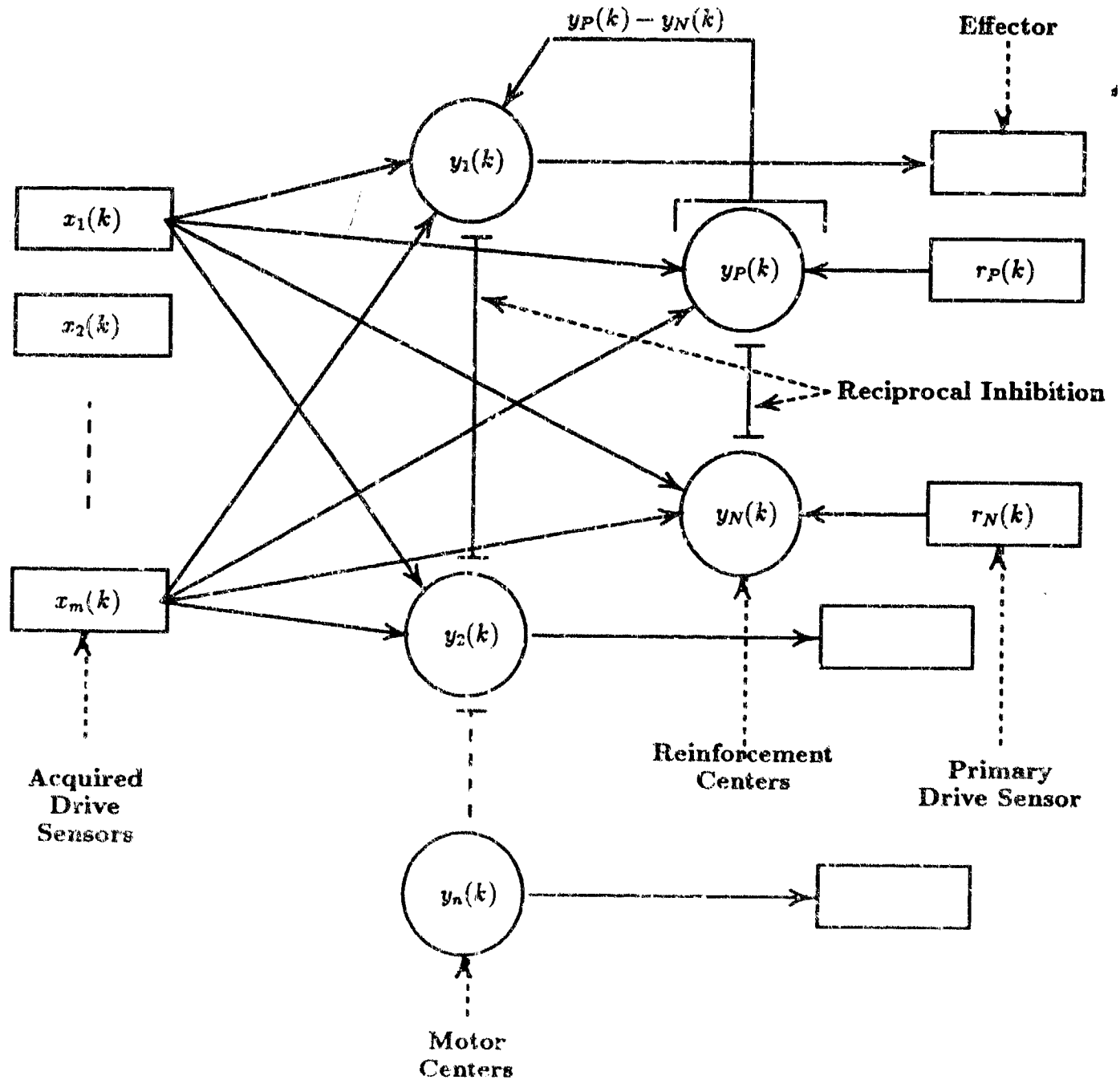


Figure 3.1. The ACP network architecture

3.1 The Original ACP

will lie within exactly one bin, with which a single acquired drive sensor is uniquely associated. The current output of the i^{th} acquired drive sensor, $x_i(k)$, will be either unity or zero, and exactly one acquired drive sensor will have unity output at each time step.⁴ The vector of m acquired drive signals, $\underline{x}(k)$, should not be confused with the vector of state variables, the length of which equals the dimension of the state space.

A motor center and effector pair exists for each discrete network output.⁵ The motor centers collectively determine the network's immediate action and, therefore, the set of n motor centers operate as a single policy center. In animal learning research, the effector encodes an action which the animal may choose to perform (e.g. to turn left). As a component of a control system, each effector represents a discrete control produced by an actuator (e.g. apply a force of 10.0 units). The output of a motor center is a real number and should not be confused with the output of the ACP network, which is an action performed by an effector.

The output of the j^{th} motor center, $y_j(k)$, equals the evaluation of a nonlinear, threshold-saturation function (Figure 3.2) applied to the weighted sum of the acquired drive sensor inputs.

$$y_j(k) = f_n \left[\sum_{i=1}^m (W_{ij}^+(k) + W_{ij}^-(k)) x_i(k) \right] \quad (3.1)$$

$$f_n(x) = \begin{cases} 0 & \text{if } x \leq \theta \\ 1 & \text{if } x \geq 1 \\ x & \text{otherwise} \end{cases} \quad (3.2)$$

⁴ This condition is not necessary in the application of the ACP to predict animal learning results.

⁵ Recall that the ACP network output must be a member of a finite set of control actions.

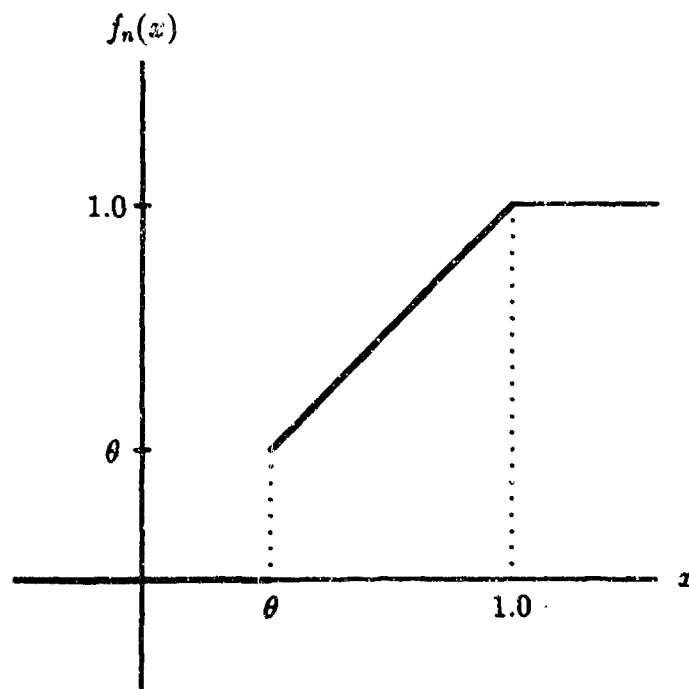


Figure 3.2. The output equation nonlinearity, (3.2).

The threshold θ is a non-negative constant less than unity. Justification for the presence of the output nonlinearity follows directly from the view that a neuronal output measures the frequency of firing of the neuron, when that frequency exceeds the neuronal threshold.⁶ Negative values of $y_j(t)$, representing negative frequencies of firing, are not physically realizable.

The motor center output equation (3.1) introduces two weights from each acquired drive sensor to each motor center: a positive *excitatory* weight $W_{ij}^+(k)$ and a negative *inhibitory* weight $W_{ij}^-(k)$. Biological evidence motivates the presence of distinct excitatory and inhibitory weights that encode attraction and avoidance

⁶ The term *neuronal output* refers to the output of a motor center or a reinforcement center.

3.1 The Original ACP

behaviors, respectively, for each state-action pair. The time dependence of the weights is explicitly shown to indicate that the weights change with time through learning; the notation does not imply that functions of time are determined for each weight.

Reciprocal inhibition, the process of comparing several neuronal outputs and suppressing all except the largest to zero, prevents the motor centers that are not responsible for the current action from undergoing weight changes. Reciprocal inhibition is defined by (3.3). The motor center $j_{max}(k)$ which wins reciprocal inhibition among the m motor center outputs at time k will be referred to as the currently active motor center; $j_{max}(k-a)$, therefore, is the motor center that was active a time steps prior to the present, and $y_{j_{max}(k-a)}(k)$ is the current output of the motor center that was active a time steps prior to the present.

$$\begin{aligned} j_{max}(k) &= j \\ \text{such that for all } l \in \{1, 2, \dots, n\} \text{ and } l \neq j \\ y_l(k) &< y_j(k) \end{aligned} \tag{3.3}$$

The current network action corresponds to the effector associated with the single motor center which has a non-zero output after reciprocal inhibition. Potentially, multiple motor centers may have equally large outputs. In this case, reciprocal inhibition for the original ACP is defined such that no motor center will be active, no control action will be effected, and no learning will occur.

The ACP architecture contains two primary drive sensors, differentiated by the labels *positive* and *negative*. The primary drive sensors provide external evaluations

Chapter 3 - The Associative Control Process

of the network's performance in the form of non-negative *reinforcement* signals; the positive primary drive sensor measures reward while the negative primary drive sensor measures cost or punishment. In the language of classical conditioning, these evaluations are collectively labeled the unconditioned stimuli. In the optimal control framework, the reward equals zero and the punishment represents an evaluation of the cost functional which the control is attempting to minimize.

The ACP architecture also contains two reinforcement centers which are identified as *positive* and *negative* and which yield non-negative outputs. Each reinforcement center learns to predict the occurrence of the corresponding external reinforcement and consequently serves as a source of internal reinforcement, allowing learning to continue in the absence of frequent external reinforcement. In this way, the two reinforcement centers direct the motor centers, through learning, to select actions such that the state approaches reward and avoids cost.

Each motor center *facilitates* a pair of excitatory and inhibitory weights from each acquired drive sensor to each reinforcement center. The output of the positive reinforcement center, prior to reciprocal inhibition between the two reinforcement centers, is the sum of the positive external reinforcement $r_P(k)$ and the weighted sum of the acquired drive sensor inputs. The appropriate set of weights from the acquired drive sensors to the reinforcement center corresponds to the currently active motor center. Therefore, calculation of the outputs of the reinforcement centers requires prior determination of $j_{max}(k)$.

$$y_P(k) = f_r \left[r_P(k) + \sum_{i=1}^m \left(W_{Pi, j_{max}(k)}^+ x_i(k) + W_{Pi, j_{max}(k)}^- x_i(k) \right) \right] \quad (3.4)$$

The output of the negative reinforcement center $y_N(k)$ is calculated similarly, using

3.1 The Original ACP

the negative external reinforcement $r_N(k)$.

$$y_N(k) = f_n \left[r_N(k) + \sum_{i=1}^m (W_{Nij_{\max}(k)}^+(k) + W_{Nij_{\max}(k)}^-(k)) x_i(k) \right] \quad (3.5)$$

The ACP learning mechanism improves the stored policy and the predictions of future reinforcements by adjusting the weights which connect the acquired drive sensors to the motor and reinforcement centers. If the j^{th} motor center is active with the i^{th} acquired drive sensor, then the reinforcement center weights $W_{Pij}^{\pm}(k)$ and $W_{Nij}^{\pm}(k)$ are eligible to change for τ subsequent time steps. The motor center weights $W_{ij}^{\pm}(k)$ are eligible to change only during the current time step.⁷ Moreover, all weights for other state-action pairs will remain constant this time step.

The impetus for motor center learning is the difference, after reciprocal inhibition, between the outputs of the positive and negative reinforcement centers. The following equations define the incremental changes in the motor center weights, where the constants c_a and c_b are non-negative. The nonlinear function f_s in (3.6), defined by (3.9), requires that only positive changes in presynaptic activity, $\Delta x_i(k)$, stimulate weight changes.

$$\Delta W_{ij}^{\pm}(k) = \begin{cases} c(k) |W_{ij}^{\pm}(k)| f_s(\Delta x_i(k)) [y_P(k) - y_N(k) - y_j(k)] & \text{if } j = j_{\max}(k) \\ 0 & \text{otherwise} \end{cases} \quad (3.6)$$

$$c(k) = c_a + c_b |y_P(k) - y_N(k)| \quad (3.7)$$

⁷ The weights of both positive and negative reinforcement centers are eligible for change even though both reinforcement centers cannot win reciprocal inhibition. In contrast, only the motor center that wins reciprocal inhibition can experience weight changes. If no motor center is currently active, however, no learning occurs in either the motor centers or the reinforcement centers.

ATTACHMENT 3

Chapter 3 - The Associative Control Process

$$\Delta x_i(k) = x_i(k) - x_i(k-1) \quad (3.8)$$

$$f_s(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

The learning process is divided into temporal intervals referred to as *trials*; the weight changes, which are calculated at each time step, are accumulated throughout the trial and implemented at the end of the trial. The symbols k_0 and k_f in (3.10) represent the times before and after a trial, respectively. A lower bound on the magnitude of every weight maintains each excitatory weight always positive and each inhibitory weight always negative (Figures 3.3 and 3.4). The constant α in (3.11) is a positive network parameter.

$$W_{ij}^+(k_f) = f_{w+} \left[W_{ij}^+(k_0) + \sum_{k=k_0}^{k_f} \Delta W_{ij}^+(k) \right] \quad (3.10a)$$

$$W_{ij}^-(k_f) = f_{w-} \left[W_{ij}^-(k_0) + \sum_{k=k_0}^{k_f} \Delta W_{ij}^-(k) \right] \quad (3.10b)$$

$$f_{w+}(x) = \begin{cases} \alpha & \text{if } x < \alpha \\ x & \text{otherwise} \end{cases} \quad (3.11a)$$

$$f_{w-}(x) = \begin{cases} -\alpha & \text{if } x > -\alpha \\ x & \text{otherwise} \end{cases} \quad (3.11b)$$

Equations (3.12) through (3.15) define the Drive-Reinforcement (DR) learning mechanism used in the positive reinforcement center; negative reinforcement center learning follows directly [12,14]. Drive-Reinforcement learning, which is a flavor of temporal difference learning [15], changes eligible connection weights as a function of the correlation between earlier changes in input signals and later changes in

ATTACHMENT 3

3.1 The Original ACP

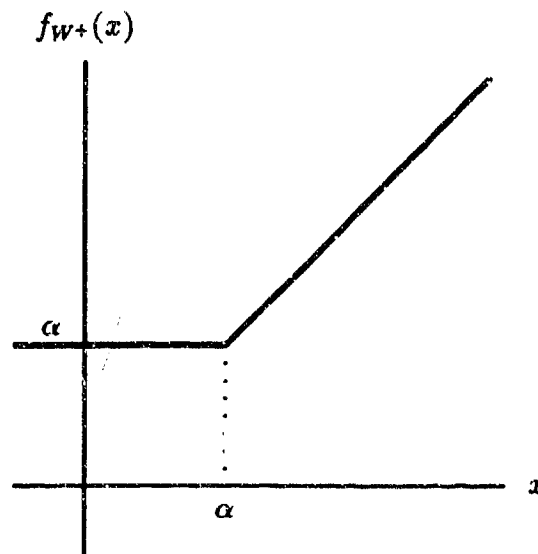


Figure 3.3. The lower bound on excitatory weights, (3.11a).

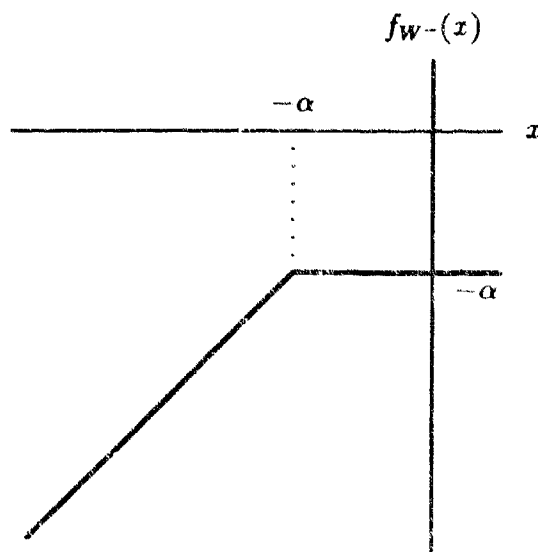


Figure 3.4. The upper bound on inhibitory weights, (3.11b).

output signals. The constants τ (which in animal learning represents the longest interstimulus interval over which delay conditioning is effective) and c_1, c_2, \dots, c_r

Chapter 3 - The Associative Control Process

are non-negative. Whereas τ may be experimentally deduced for animal learning problems, selection of an appropriate value of τ in a control problem typically requires experimentation with the particular application. The incremental change in the weight associated with a reinforcement center connection depends on four terms. The correlation between the current change in postsynaptic activity, $\Delta y_P(k)$, and a previous change in presynaptic activity, $\Delta x_i(k-a)$, is scaled by a learning rate constant c_a and the absolute value of the weight of the connection at the time of the change in presynaptic activity.

$$\Delta W_{Pij}^{\pm}(k) = \Delta y_P(k) \sum_{a=1}^{\tau} c_a |W_{Pij}^{\pm}(k-a)| f_s(\Delta x_{ij}(k-a)) \quad (3.12)$$

$$\Delta y_P(k) = y_P(k) - y_P(k-1) \quad (3.13)$$

$$\Delta x_{ij}(k-a) = \begin{cases} x_i(k-a) - x_i(k-a-1) & \text{if } j = j_{\max}(k-a) \\ 0 & \text{otherwise} \end{cases} \quad (3.14)$$

$$W_{Pij}^{+}(k_f) = f_{W+} \left[W_{Pij}^{+}(k_0) + \sum_{k=k_0}^{k_f} \Delta W_{Pij}^{+}(k) \right] \quad (3.15a)$$

$$W_{Pij}^{-}(k_f) = f_{W-} \left[W_{Pij}^{-}(k_0) + \sum_{k=k_0}^{k_f} \Delta W_{Pij}^{-}(k) \right] \quad (3.15b)$$

Note that the accumulation of weight changes until the completion of a trial eliminates the significance of the time shift in the term $|W_{Pij}^{\pm}(k-a)|$ in (3.12).

The credit assignment problem refers to the situation that some judicious choice of action at the present time may yield little or no immediate return, relative to other possible actions, but may allow maximization of future returns.⁸

⁸ The term *return* denotes a single reinforcement signal that equals the reward minus the cost. In an environment that measures simultaneous non-zero reward and cost signals, a controller should maximize the return.

3.2 Extension of the ACP

The assessment of responsibility among the recent actions for the current return is accomplished through the summation over the previous τ time steps in the reinforcement center learning equation (3.12). In the negative reinforcement center, for example, a correlation is achieved between $\Delta y_N(k)$ and the previous τ state transitions. This process of relating the current Δy_N to the previous Δx 's is referred to as chaining in animal learning. The learning rate coefficients discount the responsibility of previous actions for the current change in predicted return, where the reinforcement center outputs are predictions of future costs and rewards. Biological evidence suggests that no correlation exists between a simultaneous action and a change in predicted return, i.e. $c_0=0$, and $1 > c_j > c_\tau$ for $1 \leq j < \tau$.

3.2 Extension of the ACP to the Infinite Horizon, Optimal Control Problem

Limited modifications to the architecture and functionality of the original Associative Control Process result in a network with improved applicability to optimal control problems. Although Baird and Klopff [13] have suggested that this *modified* ACP will converge to the optimal control policy under *reasonable* assumptions, the analysis in §3.3 and the results in §3.6 suggest that the necessary conditions to obtain an optimal solution may be restrictive. This section is included to follow the development of the ACP and to motivate the single layer ACP architecture. The definition of the *modified* ACP follows from Baird and Klopff [13].

The modified ACP is applicable to a specialized class of problems; the environment with which the ACP interacts must be a non absorbing, finite state,

ATTACHMENT 3

Chapter 3 - The Associative Control Process

finite-action, discrete-time Markov decision process. Additionally, the interface between the ACP and the environment guarantees that no acquired drive sensor will exhibit unity output for more than a single consecutive time step. This stipulation results in non-uniform time steps that are artificially defined as the intervals which elapse while the dynamic state resides within a bin.⁹ The learning equations of the original ACP can be simplified by applying the fact that $x_i(k) \in \{1, 0\}$ and will not equal unity for two or more consecutive time steps. Accordingly, (3.8) and (3.9) yield,

$$f_s(\Delta x_i(k)) = \begin{cases} 1 & \text{if } x_i(k) = 1 \\ 0 & \text{otherwise.} \end{cases} \quad (3.16)$$

Therefore, a consequence of the interface between the ACP and the environment is $f_s(\Delta x_i(k)) = x_i(k)$. A similar result follows from (3.9) and (3.14).

$$f_s(\Delta x_{ij}(k-a)) = \begin{cases} 1 & \text{if } x_i(k-a) = 1 \text{ and } j = i_{\max}(k-a) \\ 0 & \text{otherwise} \end{cases} \quad (3.17)$$

The role of the reinforcement center weights becomes more well defined in the modified ACP. The sum of the inhibitory and excitatory weights in a reinforcement center estimate the expected discounted future reinforcement received if action j is performed in state i , followed by optimal actions being performed in all subsequent states. To achieve this significance, the reinforcement center output and learning equations must be recast. The external reinforcement term does not appear in the output equation of the reinforcement center; e.g. (3.4) becomes,

$$y_P(k) = f_n \left[\sum_{i=1}^m \left(W_{P, i_{\max}(k)}^+(k) + W_{P, i_{\max}(k)}^-(k) \right) x_i(k) \right]. \quad (3.18)$$

⁹ Similar to §3.1, the state space is quantized into bins.

ATTACHMENT 3

3.2 Extension of the ACP

The expression for the change in the reinforcement center output is also slightly modified. Using the example of the negative reinforcement center, (3.13) becomes,

$$\Delta y_N(k) = \gamma y_N(k) - y_N(k-1) + r_N(k) \quad \text{where } 0 < \gamma < 1. \quad (3.19)$$

If the negative reinforcement center accurately estimates the expected discounted future cost, $\Delta y_N(k)$ will be zero and no weight changes will occur. Therefore, the cost to complete the problem from time $k-1$ will approximately equal the cost accrued from time $k-1$ to k plus the cost to complete the problem from time k .¹⁰

$$y_N(k-1) = \gamma y_N(k) + r_N(k) \quad \text{when } \Delta y_N(k) = 0 \quad (3.20)$$

The value of $r_N(k)$, therefore, represents the increment in the cost functional ΔJ from time $k-1$ to k . Recall that time steps are an artificially defined concept in the modified ACP; the cost increment must be an assessment of the cost functional over the real elapsed time.¹¹ The possibility that an action selected now does not significantly effect the cost in the far future is described by the discount factor γ , which also guarantee the convergence of the infinite horizon sum of discounted future costs.

The constants in (3.7) are defined as follows: $c_a = \frac{1}{2}$ and $c_b = 0$. Additionally, the terms which involve the absolute values of the weights are removed from both the motor center learning equation and the reinforcement center learning equation.

¹⁰ This statement is strictly true for $\gamma = 1$.

¹¹ Time is discrete in this system. Time steps will coincide with an integral number of discrete time increments.

ATTACHMENT 3

Chapter 3 - The Associative Control Process

Equations (3.6) and (3.12) are written as (3.21) and (3.22), respectively. With the absence of these terms, the distinct excitatory and inhibitory weights could be combined into a single weight, which can assume positive or negative values. This change, however, is not made in [13].

$$\Delta W_{ij}^{\pm}(k) = \begin{cases} \frac{1}{2} f_s(\Delta x_i(k)) [y_P(k) - y_N(k) - y_j(k)] & \text{if } j = j_{\max}(k) \\ 0 & \text{otherwise} \end{cases} \quad (3.21)$$

$$\Delta W_{Pij}^{\pm}(k) = \Delta y_P(k) \sum_{a=1}^{\tau} c_a f_s(\Delta x_{ij}(k-a)) \quad (3.22)$$

The motor center learning equation (3.21) causes the motor center weights to be adjusted so that $W_{ij}^{+}(k) + W_{ij}^{-}(k)$ will copy the corresponding sum of weights for the reinforcement center that wins reciprocal inhibition. The saturation limits on the motor center outputs are generalized; in contrast to (3.2), $f_n(x)$ is redefined as $f_{n'}(x)$.

$$f_{n'}(x) = \begin{cases} -\beta & \text{if } x \leq -\beta \\ \beta & \text{if } x \geq \beta \\ x & \text{otherwise} \end{cases} \quad (3.23)$$

Additionally, the definition of reciprocal inhibition is adjusted slightly; the non-maximizing motor center outputs are suppressed to a minimum value $-\beta$ which is not necessarily zero.

Although the learning process is still divided into trials, the weight increments are incorporated into the weights at every time step, instead of after a trial has been completed. Equations (3.10) and (3.15) are now written as (3.24) and (3.25), respectively.

$$W_{ij}^{+}(k) = f_{w+} [W_{ij}^{+}(k-1) + \Delta W_{ij}^{+}(k)] \quad (3.24a)$$

$$W_{ij}^{-}(k) = f_{w-} [W_{ij}^{-}(k-1) + \Delta W_{ij}^{-}(k)] \quad (3.24b)$$

3.3 Motivation for the Single Layer ACP

$$W_{Pij}^+(k) = f_{W+} [W_{Pij}^+(k-1) + \Delta W_{Pij}^+(k)] \quad (3.25a)$$

$$W_{Pij}^-(k) = f_{W-} [W_{Pij}^-(k-1) + \Delta W_{Pij}^-(k)] \quad (3.25b)$$

A procedural issue arises that is not encountered in the original ACP network, where the weights are only updated at the end of a trial. The dependence of the reinforcement center outputs on $j_{max}(k)$ requires that the motor center outputs be computed first. After learning, however, the motor center outputs and also $j_{max}(k)$ may have changed, resulting in the facilitation of a different set of reinforcement center weights. Therefore, if weight changes are calculated such that $j_{max}(k)$ changes, these weight changes should be implemented and the learning process repeated until $j_{max}(k)$ does not further change this time step.

In general, exploration of the state-action space is necessary to assure global convergence of the control policy to the optimal policy, and can be achieved by occasionally randomly selecting $j_{max}(k)$, instead of following reciprocal inhibition. Initiating new trials in random states also provides exploratory information.

3.3 Motivation for the Single Layer Architecture of the ACP

This section describes qualitative observations from the application of the modified two-layer ACP to the regulation of the aeroelastic oscillator; additional quantitative results appear in §3.6. In this environment, the modified ACP learning system fails to converge to a useful control policy. This section explains the failure by illustrating several characteristics of the two-layer implementation of the ACP algorithm that are incompatible with the application to optimal control problems.

Chapter 3 - The Associative Control Process

The objective of a reinforcement learning controller is to construct a policy that, when followed, maximizes the expectation of the discounted future return. For the two-layer ACP network, the incremental return is presented as distinct cost and reward signals, which stimulate the two reinforcement centers to learn estimates of the expected discounted future cost and expected discounted future reward. The optimal policy for this ACP algorithm is to select, for each state, the action with the largest difference between estimates of expected discounted future reward and cost. However, the two-layer ACP network performs reciprocal inhibition between the two reinforcement centers and, therefore, selects the control action that either maximizes the estimate of the expected discounted future reward, or minimizes the estimate of the expected discounted future cost, depending on which reinforcement center wins reciprocal inhibition. Consider a particular state-action pair evaluated with both a large cost and a large reward. If the reward is slightly greater than the cost, only the large reward will be associated with this state-action pair. Although the true evaluation of this state-action pair is a small positive return, this action in this state may be incorrectly selected as optimal.

The reinforcement center learning mechanism incorporates both the current and the previous outputs of the reinforcement center. For example, the positive reinforcement center learning equation includes the term $\Delta y_P(k)$, given in (3.26), which represents the error in the estimate of the expected discounted future reward for the previous state $y_P(k-1)$.

$$\Delta y_P(k) = \gamma y_P(k) - y_P(k-1) + r_I(k) \quad (3.26)$$

A reinforcement center that loses the reciprocal inhibition process will have an out-

3.3 Motivation for the Single Layer ACP

put equal to zero. Consequently, the value of $\Delta y_P(k)$ will not accurately represent the error in $y_P(k-1)$ when $y_P(k)$ or $y_P(k-1)$ equals zero as a result of reciprocal inhibition. Therefore, $\Delta y_P(k)$ will be an invalid contribution to reinforcement learning if the positive and negative reinforcement centers alternate winning reciprocal inhibition. Similarly, $\Delta y_N(k)$ may be erroneous by a parallel argument. Moreover, the fact that learning occurs even for the reinforcement center which loses reciprocal inhibition assures that either $\Delta y_P(k)$ or $\Delta y_N(k)$ will be incorrect on every time step that a motor center is active. If no motor center is active, no set of weights between the acquired drive sensors and reinforcement centers are facilitated and both reinforcement centers will have zero outputs. Although no learning occurs in the reinforcement centers on this time step, both Δy_P and Δy_N will be incorrect on the next time step that a motor center is active.

The difficulties discussed above, which arise from the presence of two competing reinforcement centers, are reduced by providing a non-zero external reinforcement signal to only a single reinforcement center. However, the reinforcement center which receives zero external reinforcement will occasionally win reciprocal inhibition until it learns that zero is the correct output for every state. Using the sum of the reinforcement center output and the external reinforcement signal as the input to the reciprocal inhibition process may guarantee that a single reinforcement center will always win reciprocal inhibition.¹²

The optimal policy for each state is defined by the action which yields the largest expected discounted future return. The ACP network represents this in-

¹² The original ACP uses this technique in (3.4) and (3.5); the modified two-layer ACP eliminates the external reinforcement signal from the reinforcement center output in (3.18).

ATTACHMENT 3

Chapter 3 - The Associative Control Process

formation in the reinforcement centers and, through learning, transfers the value estimates to the motor centers, where an action is selected through reciprocal inhibition. The motor center learning mechanism copies either the estimate of expected discounted future cost or the estimate of expected discounted future reward, depending on which reinforcement center wins reciprocal inhibition, into the single currently active motor center for a given state. Potentially, each time this state is visited, a different reinforcement center will win reciprocal inhibition and a different motor center will be active. Therefore, at a future point in time, when this state is revisited, reciprocal inhibition between the motor center outputs may compare estimates of expected discounted future cost with estimates of expected discounted future reward. This situation, also generated when the two reinforcement centers alternate winning reciprocal inhibition, invalidates the result of reciprocal inhibition between motor centers. Therefore, the ACP algorithm to select a policy does not guarantee that a complete set of estimates of a consistent evaluation (i.e. reward, cost, or return) will be compared over all possible actions.

This section has introduced several fundamental limitations in the two-layer implementation of the ACP algorithm, which restrict its applicability to optimal control problems. By reducing the network to a single layer of learning centers, the resulting architecture does not interfere with the operation of the Drive-Reinforcement concept to solve infinite-horizon optimization problems.

3.4 A Single Layer Formulation of the ACP

3.4 A Single Layer Formulation of the Associative Control Process

The starting point for this research was the original Associative Control Process. However, several elements present in the original ACP network, which are consistent with the known physiology of biological neurons, are neither appropriate nor necessary in a network solely intended as an optimal controller. This section presents a single layer formulation of the modified ACP (Figure 3.5), and contains significantly fewer adjustable parameters, fewer element types, and no nonlinearity in the output equation. Although the physical structure of the single layer network is not faithful to biological evidence, the network retains the ability to predict classical and instrumental conditioning results [13].

The interface of the environment to the single layer network through m input sensors is identical to the interface to the modified ACP network through the acquired drive sensors. A single external reinforcement signal $r(k)$, which assesses the incremental return achieved by the controller's actions, replaces the distinct reward and cost external reinforcement signals present in the two-layer network.

A *node* and effector pair exists for each discrete network action.¹³ The output of the j^{th} node estimates the expected discounted future return for performing action j in the current state and subsequently following an optimal policy. The sum of an excitatory and an inhibitory weight encode this estimate. Constructed from a single type of neuronal element, the single layer ACP architecture requires

¹³ A *node* combines the functionality of the motor and reinforcement centers.

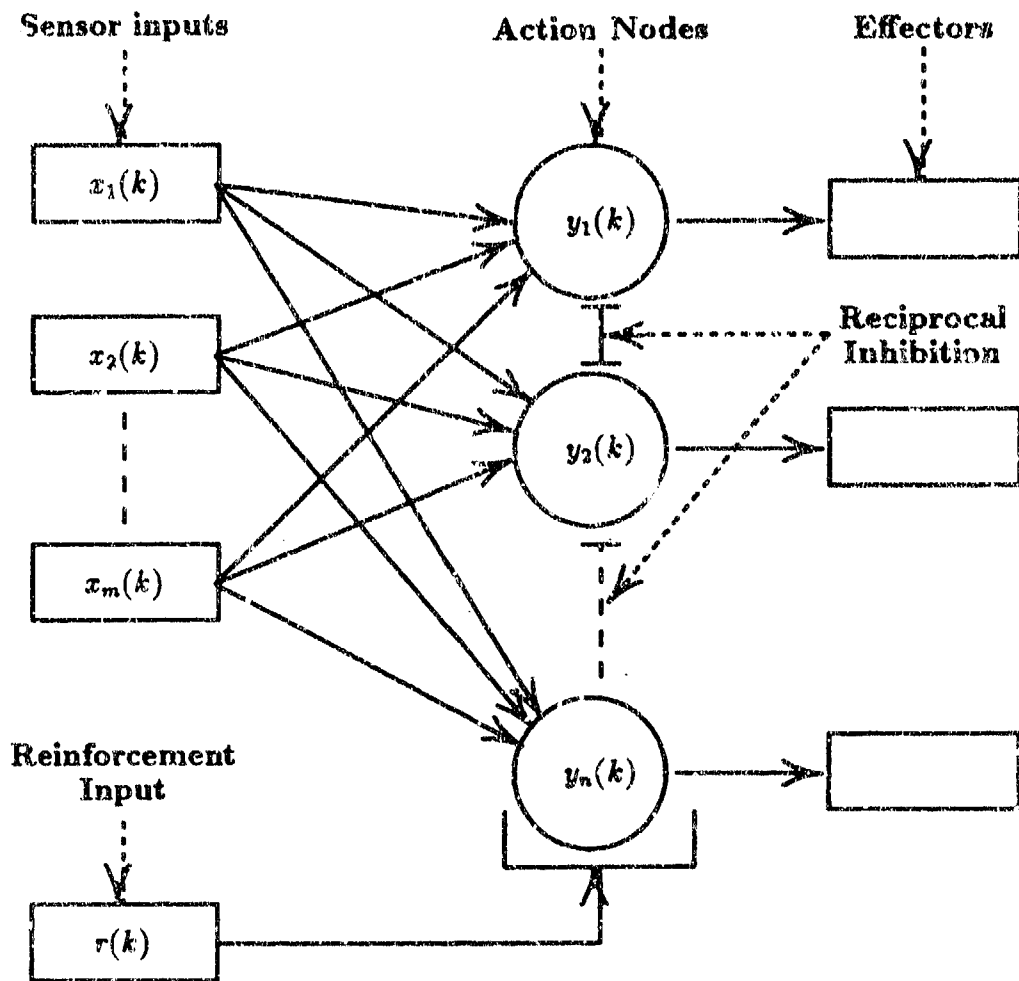


Figure 3.5. The single layer ACP architecture.

only a single linear output equation and a single learning equation.

$$y_i(k) = \sum_{i=1}^m (W_{ij}^+(k) + W_{ij}^-(k)) x_i(k) \quad (3.27)$$

The optimal policy, to maximize the expected discounted future return, selects for each state the action corresponding to the node with greatest output. Reciprocal inhibition between the n nodes defines a currently active node $j_{max}(k)$, similar to

3.5 Implementation

the process between motor centers in the two-layer ACP. However, the definition of reciprocal inhibition has been changed in the situation where multiple nodes have equally large outputs. In this case, which represents a state with multiple equally optimal actions, $j_{max}(k)$ will equal the node with the smallest index j . Therefore, the controller will perform an action and will learn on every time step.

The learning equation for a node resembles that of a reinforcement center. However, the absolute value of the connection weight at the time of the state change, which was removed in the modified ACP, has been restored into the learning equation [13]. This term, which was originally introduced for biological reasons, is not essential in the network and serves as a learning rate parameter. The discount factor γ describes how an assessment of return in the future is less significant than an assessment of return at the present. As before, only weights associated with a state-action pair being active in the previous τ time steps are eligible for change.

$$\Delta W_{ij\pm}(k) = \left[\gamma y_{j_{max}(k)}(k) - y_{j_{max}(k-1)}(k-1) + r(k) \right] \cdot \sum_{a=1}^{\tau} c_a |W_{ij\pm}(k-a)| f_s(\Delta x_{ij}(k-a)) \quad (3.28)$$

$$f_s(\Delta x_{ij}(k-a)) = \begin{cases} 1 & \text{if } j = j_{max}(k-a) \text{ and } x_i(k-a) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (3.29)$$

$$W_{ij}^{+}(k) = f_{W^{+}} \left[W_{ij}^{+}(k-1) + \Delta W_{ij}^{+}(k) \right] \quad (3.30a)$$

$$W_{ij}^{-}(k) = f_{W^{-}} \left[W_{ij}^{-}(k-1) + \Delta W_{ij}^{-}(k) \right] \quad (3.30b)$$

3.5 Implementation

The modified two-layer ACP algorithm and the single layer ACP algorithm were implemented in *NetSim* and evaluated as regulators of the AEO plant; fundamental limitations prevented a similar evaluation of the original ACP algorithm. The experiments discussed in this section and in §3.6 were not intended to represent an exhaustive analysis of the ACP methods. For several reasons, investigations focused more heavily on the Q learning technique, to be introduced in §4. First, the ACP algorithms can be directly related to the Q learning algorithm. Second, the relative functional simplicity of Q learning, which also possesses fewer free parameters, facilitated the analysis of general properties of direct learning techniques applied to optimal control problems.

This section details the implementation of the ACP reinforcement learning algorithms. The description of peripheral features that are common to both the ACP and Q learning environments will not be repeated in §4.5.

The requirement that the learning algorithm's input space consist of a finite set of disjoint states necessitated a BOXES [8] type algorithm to quantize the continuous dynamic state information that was generated by the simulation of the AEO equations of motion.¹⁴ As a result, the input space was divided into 200 discrete states. The 20 angular boundaries occurred at 18° intervals, starting at 0°; the 9 boundaries in magnitude occurred at 1.15, 1.0, 0.85, 0.7, 0.55, 0.4, 0.3, 0.2, and 0.1

¹⁴ The terms *bins* and *discrete states* are interpreted synonymously. The aeroelastic oscillator has two state variables: position and velocity. The measurement of these variables in the space of continuous real numbers will be referred to as the *dynamic state* or *continuous state*.

3.5 Implementation

nondimensional units; the outer annulus of bins did not have a maximum limit on the magnitude of the state vectors that it contained.

The artificial definition of time steps as the non-uniform intervals between entering and leaving bins eliminates the significance of τ as the longest interstimulus interval over which delay conditioning is effective.

The ACP learning control system was limited to a finite number of discrete outputs: $+0.5$ and -0.5 nondimensional force units.

The learning algorithm operated through a hierarchical process of *trials* and *experiments*. Each experiment consisted of numerous trials and began with the initialization of weights and counters. Each trial began with the random initialization of the state variables and ran for a specified length of time.¹⁵ In the two-layer architecture, the motor center and reinforcement center weights were randomly initialized using uniform distributions between $\{-1.0, -\alpha\}$ and $\{\alpha, 1.0\}$. In the single layer architecture, all excitatory weights were initialized within a small uniform random deviation of 1.0, and all inhibitory weights were initialized within a small uniform random deviation of $-\alpha$. The impetus for this scheme was to originate weights sufficiently large such that learning with non-positive reinforcement (i.e. zero reward and non-negative cost) would only decrease the weights.

The learning system operates in discrete time. At every time step, the dynamic state transitions to a new value either in the same bin or in a new bin and the system evaluates the current assessment of either cost and reward or reinforcement. For each discrete time step that the state remains in a bin, the reinforcement

¹⁵ Initial states (position and velocity) were uniformly distributed between -1.2 and $+1.2$ nondimensional units.

ATTACHMENT 3

Chapter 3 - The Associative Control Process

Table 3.1. ACP parameters.

| Name | Symbol | Value |
|--------------------------------|----------|-------|
| Discount Factor | γ | 0.95 |
| Threshold | θ | 0.0 |
| Minimum Bound on $ W $ | α | 0.1 |
| Maximum Motor Center Output | β | 1.0 |
| Maximum Interstimulus Interval | τ | 5 |

accumulates as the sum of the current reinforcement and the accretion of previous reinforcements discounted by γ . The arrows in Figure 3.6 with arrowheads lying in Bin 1 represent the discrete time intervals that contribute reinforcement to learning in Bin 1. Learning for Bin 1 occurs at t_5 where the total reinforcement equals the sum of r_5 and γ times the total reinforcement at t_4 .

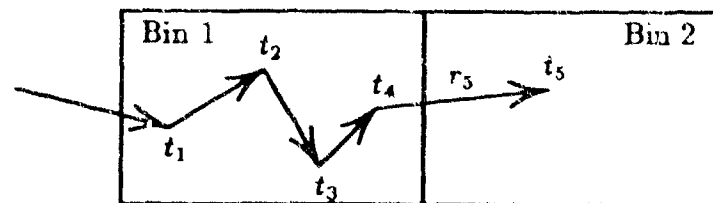


Figure 3.6. A state transition and reinforcement accumulation cartoon.

For the two-layer ACP, the reward presented to the positive reinforcement center was zero, while the cost presented to the negative reinforcement center was a quadratic evaluation of the state error. In the single layer learning architecture,

3.6 Results

the quadratic expression for the reinforcement signal r , for a single discrete time interval, was the negative of the product of the square of the magnitude of the state vector, at the final time for that interval, and the length of that time interval. The quadratic expression for cost in the two-layer ACF was $-r$. The magnitude of the control expenditure was omitted from the reinforcement function because the contribution was constant for the two-action control laws.

$$r = -(t_2 - t_3) [x(t_2)^2 + \dot{x}(t_2)^2] \quad (3.31)$$

3.6 Results

Figure 3.7 illustrates a typical segment of a trial, prior to learning, in which an ACP learning system regulated the AEO plant; the state trajectory wandered clockwise around the phase plane, suggesting the existence of two stable limit cycles.

The modified two-layer ACP system failed to learn a control law which drove the state from an arbitrary initial condition to the origin. Instead, the learned control law produced trajectories with unacceptable behavior near the origin (Figure 3.8). The terminal condition for the AEO state controlled by an optimal regulator with a finite number of discrete control levels, is a limit cycle. However, the two-layer ACP failed to converge to the optimal control policy. Although the absence of a set of learning parameters for which the algorithm would converge to an optimal solution cannot be easily demonstrated, §3.3 clearly identifies several undesirable properties of the algorithm.

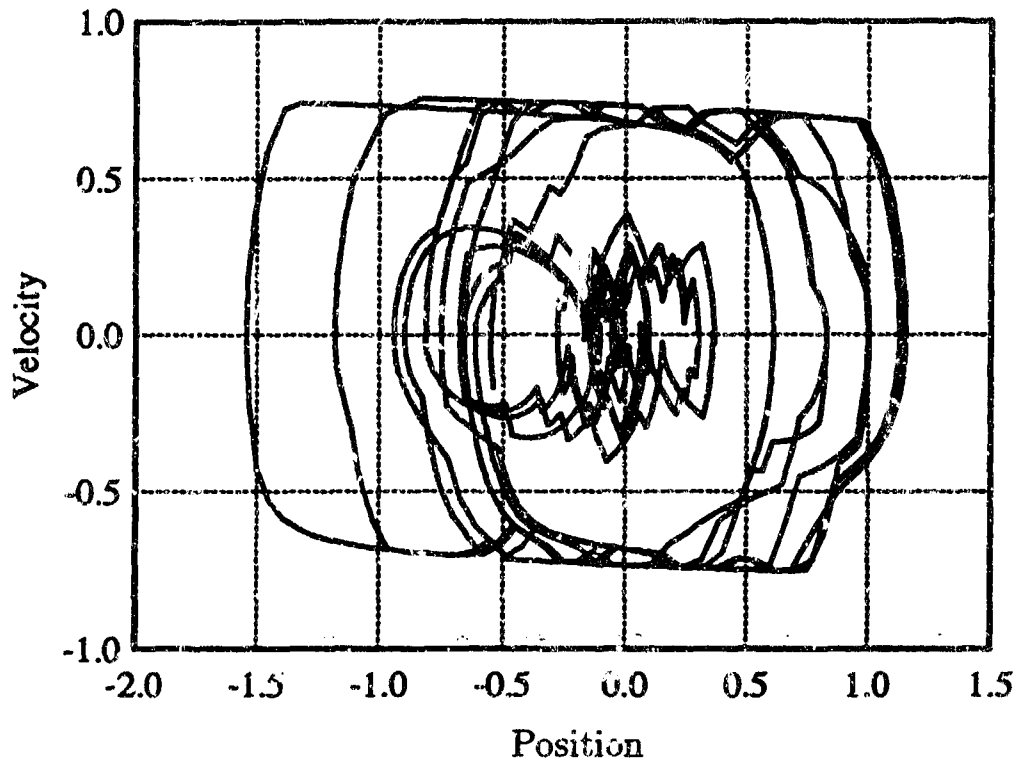


Figure 3.7. A characteristic AEO state trajectory achieved by a reinforcement learning algorithm prior to learning.

The single layer architecture of the ACP learned the optimal control law, which successfully regulated the AEO state variables near zero from any initial condition within the region of training, $\{-1.2, 1.2\}$. The performance of the control policy was limited by the coarseness of the bins and the proximity of bin boundaries to features of the nonlinear dynamics. The restricted choice of control actions also bounds the achievable performance, contributing to the rough trajectory in Figure 3.9.

ATTACHMENT 3

3.8 Results

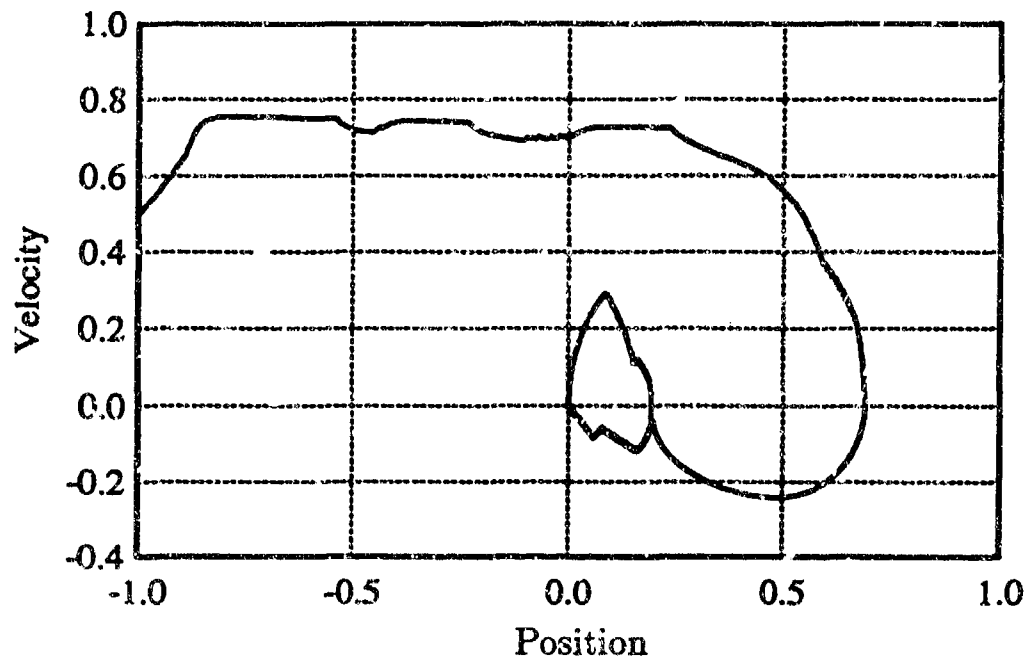


Figure 3.8. The AEO state trajectory achieved by the modified two-layer ACP after learning.

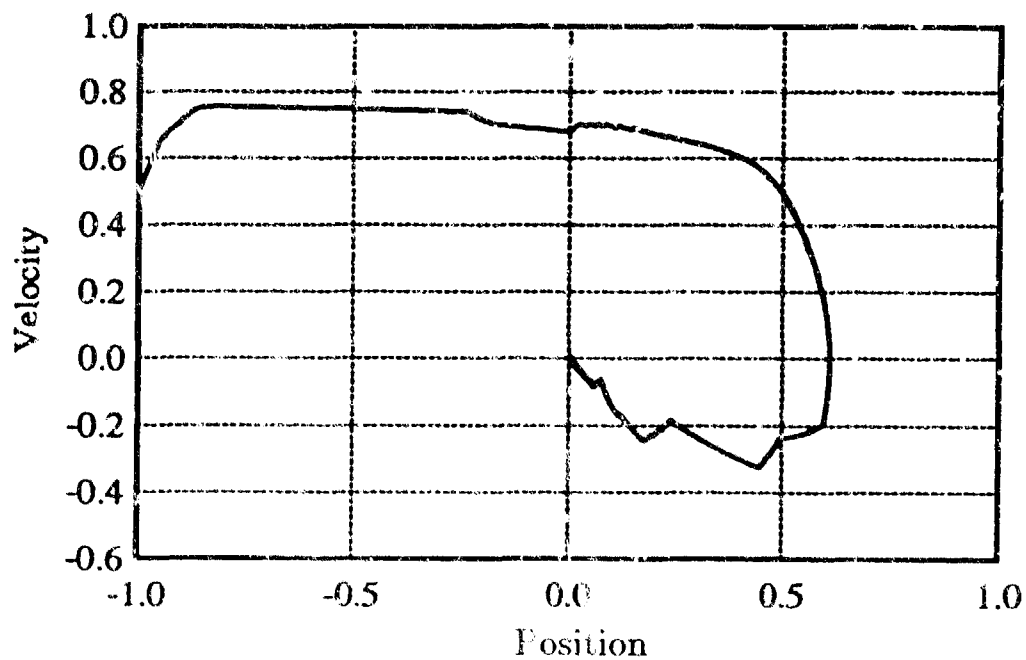


Figure 3.9. The AEO state trajectory achieved by the single layer ACP after learning.

Chapter 4

Policy and Value Iteration

The notation and concepts presented in §4.1 through §4.4 follow directly from Watkins' thesis [16] and [17]. §4.5 and §4.6 present results of applying Q learning to the AEO. §4.7 explores a continuous version of Q learning.

4.1 Terminology

4.1.1 Total Discounted Future Return

A discrete-time system that performs an action a_k in a state x_k , at time k , receives a performance evaluation r_k associated with the transition to the state x_{k+1} at time $k + 1$; the evaluation r_k is referred to as the return at time k .¹ The *total future return* after time k , which equals the sum of the returns assessed between time k and the completion of the problem, may be unbounded for an infinite

¹ Watkins defines *return* as the total discounted future *reward*; this paper equates the terms *return* and *reward*.

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

horizon problem. However, the return received in the distant future is frequently less important, at the present time, than contemporary evaluations. Therefore, the *total discounted future return*, defined in (4.1) and guaranteed to be finite, is proposed.

$$\sum_{n=0}^{\infty} \gamma^n r_{k+n} = r_k + \gamma r_{k+1} + \gamma^2 r_{k+2} + \dots + \gamma^n r_{k+n} + \dots \quad (4.1)$$

The discount factor, $0 \leq \gamma < 1$, determines the present value of future returns.

4.1.2 The Markov Decision Process

A non-absorbing, finite-state, finite-action, discrete time Markov decision process is described by a bounded set of states S , a countable set of actions for each state $A(x)$ where $x \in S$, a transition function T , and an evaluation mechanism R . At time k , the state is designated by a random variable X_k and the true value x_k . The transition function defines $X_{k+1} = T(x_k, a_k)$ where $a_k \in A(x_k)$; the new state must not equal the previous state with probability equal to unity. At time k , the return is denoted by a random variable $R_k = R(x_k, a_k)$ and the actual evaluation r_k . The expectation of the return is written \bar{R}_k . The Markov property implies that the transition and evaluation functions depend on the current state and current action, and do not depend on previous states, actions, or evaluations.

4.1.3 Value Function

In a Markov decision process, the expectation of the total discounted future return depends only on the current state and the stationary policy. A convenient notation for the probability that performing action a in state x will leave the

ATTACHMENT 3

4.1 Terminology

system in state y is $P_{xy}(a)$. The random variable representing the future state X_{k+n} achieved by starting in the state x_k at time k and following policy f for n time steps is written as $X(x_k, f, n)$.

$$X(x_k, f, 0) = x_k \quad (4.2a)$$

$$X(x_k, f, 1) = X_{k+1} = T(x_k, f(x_k)) \quad (4.2b)$$

If policy f is followed for n time steps from state x_k at time k , the return realized for applying $f(x_{k+n})$ in state x_{k+n} is expressed as $R(x_k, f, n)$.

$$R(x_k, f, 0) = R(x_k, f(x_k)) = R_k \quad (4.3a)$$

$$R(x_k, f, n) = R(X_{k+n}, f(X_{k+n})) = R_{k+n} \quad (4.3b)$$

The expected total discounted future return subsequent to the state x , applying the invariant policy f , is the *value function* $V_f(x)$.

$$V_f(x) = \overline{R(x, f, 0) + \gamma R(x, f, 1) + \dots + \gamma^n R(x, f, n) + \dots} \quad (4.4a)$$

$$= \overline{R(x, f, 0) + \gamma V_f(X(x, f, 1))} \quad (4.4b)$$

$$= \overline{R(x, f, 0) + \gamma \sum_y P_{xy}(f(x)) V_f(y)} \quad (4.4c)$$

In (4.4c), y is the subset of S that is reachable from x in a single time step.

4.1.4 Action Value

The *action-value* $Q_f(x, a)$ is the expectation of the total discounted future return for starting in state x , performing action a , and subsequently following

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

policy f . Watkins refers to action-values as Q values. A Q value represents the same information as the sum of an excitatory weight and an inhibitory weight in Drive-Reinforcement learning, which is used in the single layer ACP.

$$Q_f(x, a) = \overline{R(x, a)} + \gamma \sum_y P_{xy}(a) V_f(y) \quad (4.5)$$

The expression for an action-value (4.5) indicates that the value function for policy f must be completely known prior to computing the action-values.

Similarly, $Q_f(x, g)$ is the expected total discounted future return for starting in x , performing action $g(x)$ according to policy g , and subsequently following policy f .

4.2 Policy Iteration

The *Policy Improvement Theorem* [16] states that a policy g is uniformly better than or equivalent to a policy f if and only if,

$$Q_f(x, g) \geq V_f(x) \quad \text{for all } x \in S. \quad (4.6)$$

This theorem and the definition of action-values imply that for a policy g which satisfies (4.6), $V_g(x) \geq V_f(x)$ for all $x \in S$. The *Policy Improvement Algorithm* selects an improved policy g according to the rule: $g(x) = a \in A(x)$ such that a is the argument that maximizes $Q_f(x, a)$. However, to determine the action-values $Q_f(x, a)$ for f , the entire value function $V_f(x)$ must first be calculated. In

ATTACHMENT 3

4.3 Value Iteration

the context of a finite-state, finite-action Markov process, policy improvement will terminate after applying the algorithm a finite number of times; the policy g will converge to an optimal policy.

The *Optimality Theorem* [16] describes a policy f^* which cannot be improved using the policy improvement algorithm. The associated value function $V_{f^*}(x)$ and action-values $Q_{f^*}(x, a)$ satisfy (4.7) and (4.8) for all $x \in S$.

$$V_{f^*}(x) = \max_{a \in A(x)} Q_{f^*}(x, a) \quad (4.7)$$

$$f^*(x) = a \quad \text{such that} \quad Q_{f^*}(x, a) = V_{f^*}(x) \quad (4.8)$$

The optimal value function and action-values are unique; the optimal policy is unique except in states for which several actions yield equal and maximizing action-values.

4.3 Value Iteration

The *value iteration* [16] procedure calculates an optimal policy by choosing for each state the action which effects a transition to the new state that possesses the maximum evaluation; the optimal value function determines the evaluation of each state that succeeds the current state. The expected total discounted future return for a finite horizon process which consists of n transitions and a subsequent final return, to evaluate the terminal state, is represented as V^n . The value function

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

which corresponds to the infinite horizon problem, is approximated by repeatedly applying rule (4.9) to an initial estimate V^0 .

$$V^n(x) = \max_{a \in A(x)} \left[\overline{R(x, a)} + \gamma \sum_y P_{xy}(a) V^{n-1}(y) \right] \quad (4.9)$$

Value iteration guarantees that the limit in (4.10) approaches zero uniformly over all states. Therefore, V^n converges to the optimal value function and the optimal policy can be derived directly from V_f^* .

$$\lim_{n \rightarrow \infty} |V^n - V_f^*| = 0 \quad (4.10)$$

Although this procedure is computationally simplest if all states are systematically updated so that V^n is completely determined from V^{n-1} before V^{n+1} is calculated for any state, Watkins has demonstrated that the value iteration method will still converge if the values of individual states are updated in an arbitrary order, provided that all states are updated sufficiently frequently.

4.4 Q Learning

Unfortunately, neither the optimal policy nor optimal value function can be initially known in a control problem. Therefore, the learning process involves simultaneous, incremental improvements in both the policy function and the value function. Action-values $Q_{f_k}(x_k, a_k)$ for each state-action pair at time k contain

4.5 Implementation

both policy and value information; the policy and value functions at time k are defined in (4.11) and (4.12) in terms of Q values.

$$f_k^Q(x) = a \quad \text{such that} \quad Q_k(x, a) = V_k^Q(x) \quad (4.11)$$

$$V_k^Q(x) = \max_a [Q_k(x, a)] \quad (4.12)$$

The superscript Q denotes the derivation of the policy and the value function from the set of action-values $Q_{f_k}(x_k, a_k)$. Single step Q learning adjusts the action-values according to (4.13).

$$Q_{k+1}(x_k, a_k) = (1 - \alpha)Q_k(x_k, a_k) + \alpha(r_k + \gamma V_k^Q(x_{k+1})) \quad (4.13)$$

The positive learning rate constant α is less than unity. Only the action-value of the state-action pair (x_k, a_k) is altered at time k ; to guarantee convergence of the value function to the optimal, each action must be repeatedly performed in each state. As a form of dynamic programming, Q learning may be described as incremental Monte-Carlo value iteration.

4.5 Implementation

This section formalizes the implementation of the Q learning algorithm as a regulator for the aeroelastic oscillator plant. The environment external to the Q learning process was similar to that used for the ACP experiments in §3.5 and §3.6.

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

However, the quantization of the state space was altered. The boundaries of the 260 bins that covered the state space were defined by magnitudes M and angles A ; the outer annulus of bins did not have a maximum magnitude.

$$M = \{0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.6,$$

$$0.7, 0.85, 1.0\}$$

$$A = \{0^\circ, 18^\circ, 36^\circ, 54^\circ, 72^\circ, 90^\circ, 108^\circ, 126^\circ, 144^\circ, 162^\circ, 180^\circ, 198^\circ,$$

$$216^\circ, 234^\circ, 252^\circ, 270^\circ, 288^\circ, 306^\circ, 324^\circ, 342^\circ\}$$

The bins were labeled with integer numbers from 0 to 259, starting with the bins in the outer ring, within a ring increasing in index with increasing angle from 0° , and continuing to the next inner ring of bins.

For each state-action pair, the Q learning algorithm stores a real number that represents the Q value. At the start of a new *NetSim* experiment, all Q values were initialized to zero.

The two parameters which appear in (4.13) were: $\gamma = 0.95$ and $\alpha = 0.5$. In this context, α is a learning rate parameter; in the ACP description, α was the minimum bound on the absolute value of the weights. The return r_k was given in (3.31) as the negative of the product of the squared magnitude of the state vector and the length of the time interval.

4.6 Results

The results of two experiments, conducted in the *NetSim* [11] environment, characterize the performance of the Q learning algorithm. The two experiments

ATTACHMENT 3

4.6 Results

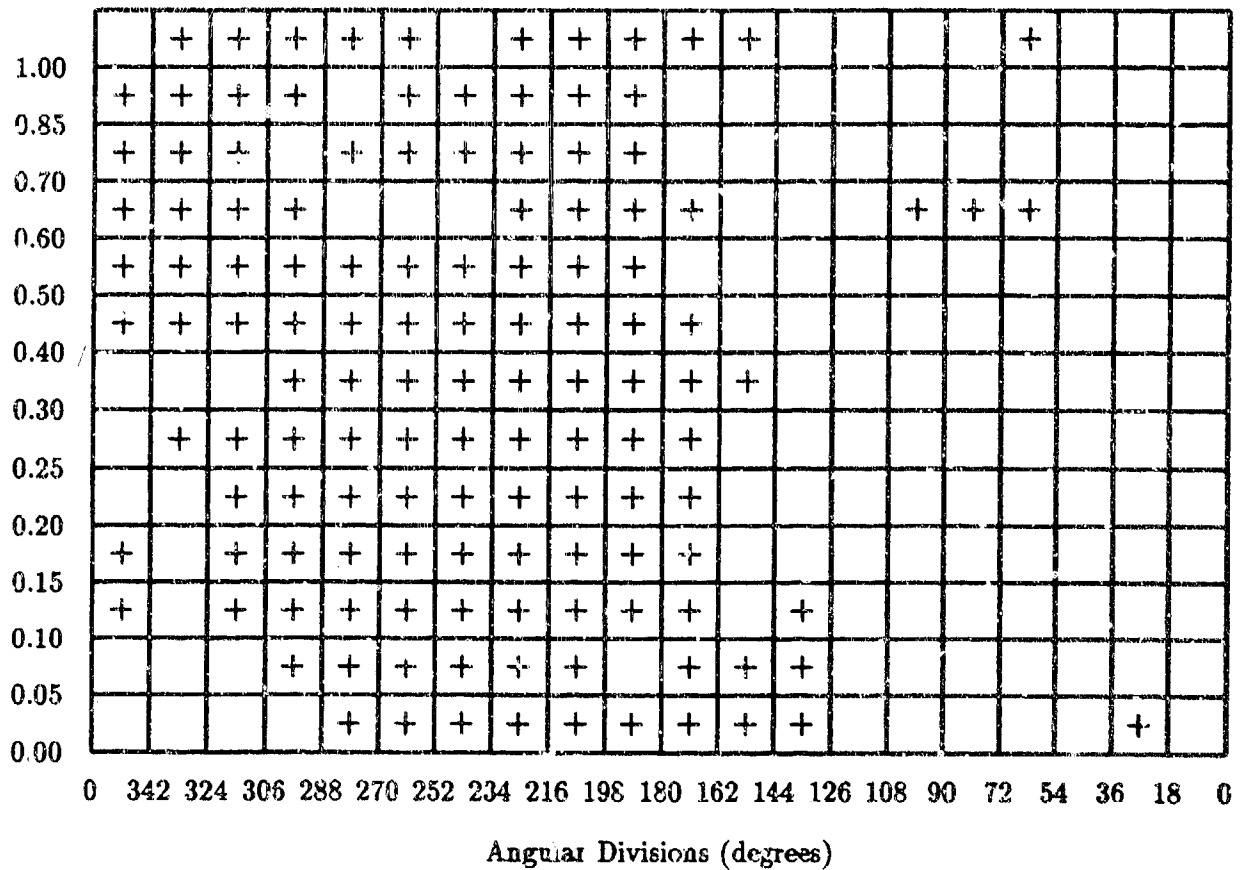


Figure 4.1a. A Cartesian representation of the two-action optimal control policy.

differ in the set of allowable control actions.

Experiment 1: $u_k \in \{0.5, -0.5\}$

Experiment 2: $u_k \in \{0.5, 0.33, 0.167, 0.0, -0.167, -0.33, -0.5\}$

The learned optimal policy for *Experiment 1* appears in Figures 4.1a and 4.1b. The control law applied a +0.5 force whenever the state resided in a bin containing a + and applied -0.5 whenever the state was in an empty bin. The general form of this control policy resembles the non-optimal bang-bang law that was derived from

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

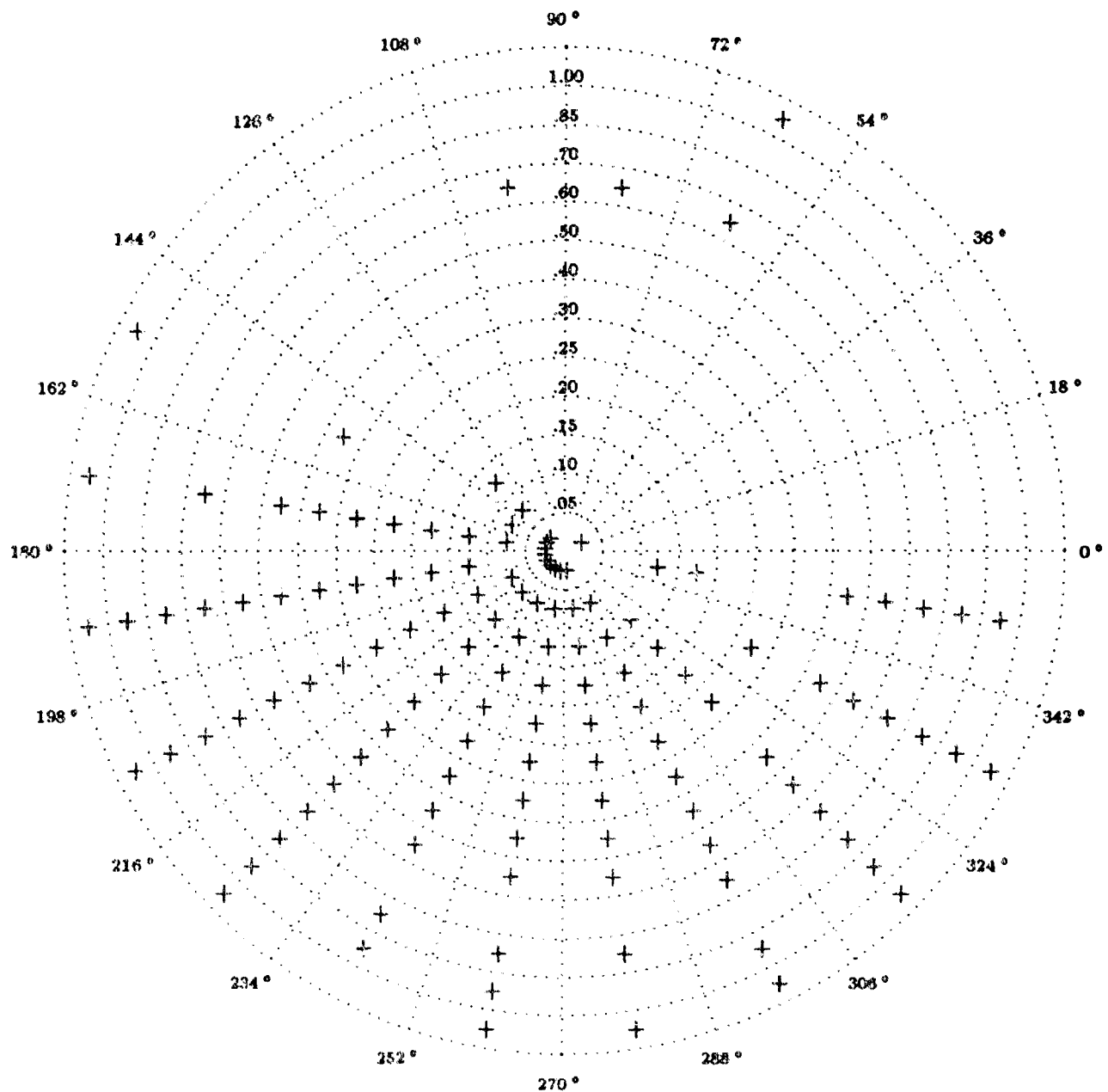


Figure 4.1b. A polar representation of the two-action optimal control policy.

a LQR solution in §2.4.3. Figure 2.9 demonstrated that for the non-optimal bang-bang control policy, the state trajectory slowly approached the origin along a linear

ATTACHMENT 3

4.6 Results

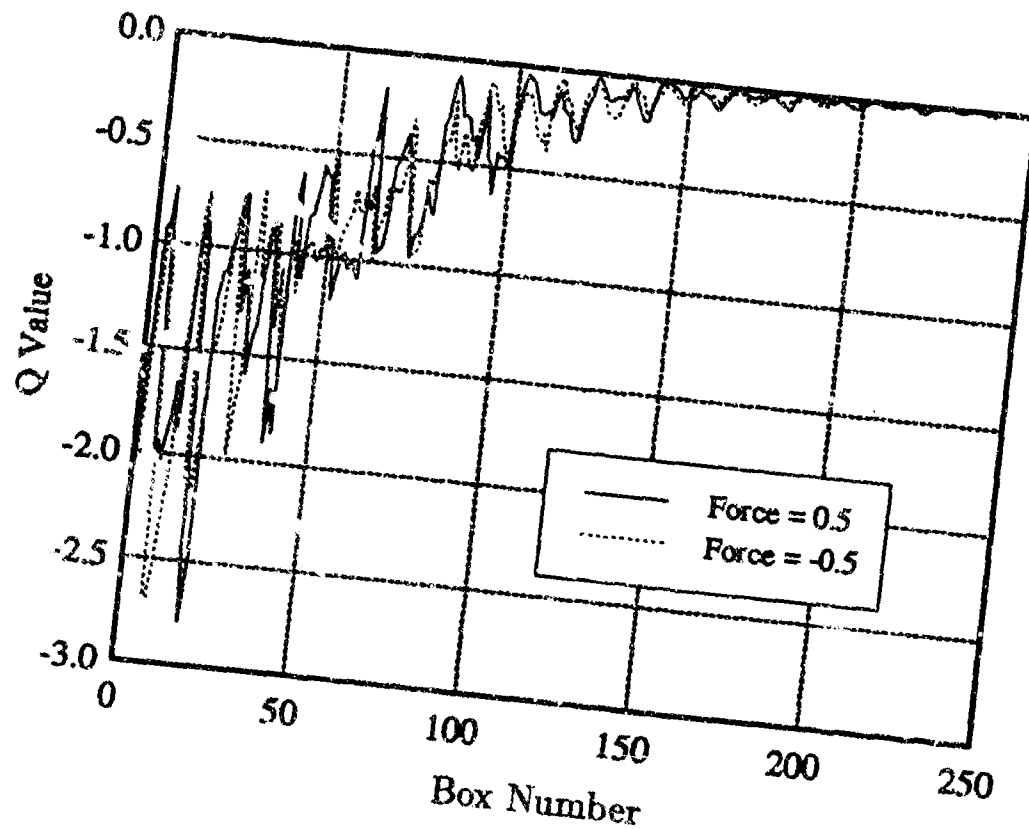


Figure 4.2. *Experiment 1:* Expected discounted future return (Q value) for each state-action pair.

switching curve. To avoid the high cost of this behavior, the optimal two-action solution will not contain a linear switching curve. A positive force must be applied in some states where the bang-bang rule (§2.4.3) dictated a negative force and a negative force must be applied in some bins below the linear switching curve. The trajectories that result from the control policies constructed in *Experiments 1* and *2* avoid the problem of slow convergence along a single switching curve. Although some regions of the control policy appear to be arbitrary, there exists a structure. For two bins bounded by the same magnitudes and separated by 180° , the optimal actions will typically be opposite. For example, the three $+$ bins bounded by 0.6, 0.7, 54° , and 108° are reflections of the blank bins bounded by 0.6, 0.7, 234° , and

Chapter 4 - Policy and Value Iteration

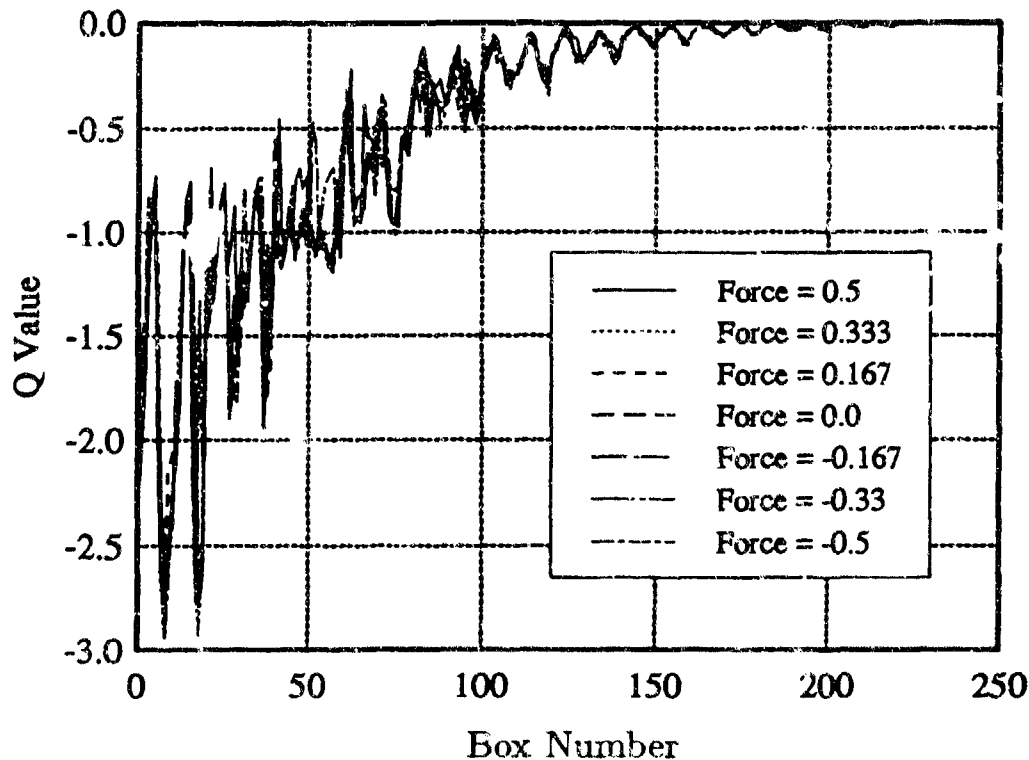


Figure 4.3. *Experiment 2:* Expected discounted future return (Q value) for each state-action pair.

288°. The 15 pairs of bins which violate this pattern lie primarily near the linear switching curve (§2.4.3).

Figures 4.2 and 4.3 compare the expected discounted future returns for all state-action pairs in *Experiments 1* and *2*, respectively. The expected discounted future return was negative for all state action pairs because only negative return (i.e. cost) was assessed. Moreover, the Q values for bins nearer the origin were greater than the Q values for outlying bins. The fact that a non-optimal action performed in a single bin does not significantly affect the total cost for a trajectory, when optimal actions are followed in all other bins (in this problem), explains the similarity between most Q values associated with different actions and the same

4.6 Results

state. Additionally, the Q values varied almost periodically as a function of the bin number; the largest variance existed for the bins farthest from the origin. All trajectories tended to approach the origin along the same paths through the second and fourth quadrants (Figures 4.4, 4.6, 4.8, and 4.10). Therefore, if an initial condition was such that the limited control authority could not move the state onto the nearest path toward the origin, then the trajectory circled halfway around the state space to the next path toward the origin. This characteristic was a property of the AEO nonlinear dynamics, and accounted for the large differences in Q values for neighboring bins. In *Experiment 1*, there existed bins for which the choice of the initial control action determined whether this circling was necessary. For these bins, the expected discounted future returns for the two actions differed substantially.

The control law constructed in *Experiment 2* was expected to outperform the control law constructed in *Experiment 1* (i.e. for each bin, the maximum Q value from Figure 4.3 would exceed the maximum Q value from Figure 4.2). For the data presented, this expectation is true for 60% of the bins. The bins that violate this prediction are entirely located in the regions of the state space that the state enters least frequently. *Experiment 2*, with a greater number of state-action pairs, requires substantially more training than *Experiment 1*. The fact that for certain bins, the maximum Q value from *Experiment 1* exceeds that for *Experiment 2* signals insufficient learning in those bins for *Experiment 2*.

No explicit search mechanism was employed during learning. Moreover, the dynamics tended to force all trajectories onto the same paths, so that many bins were seldom entered. Therefore, to assure that a globally optimal policy was attained, sufficient trials were required so that the random selection of the initial

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

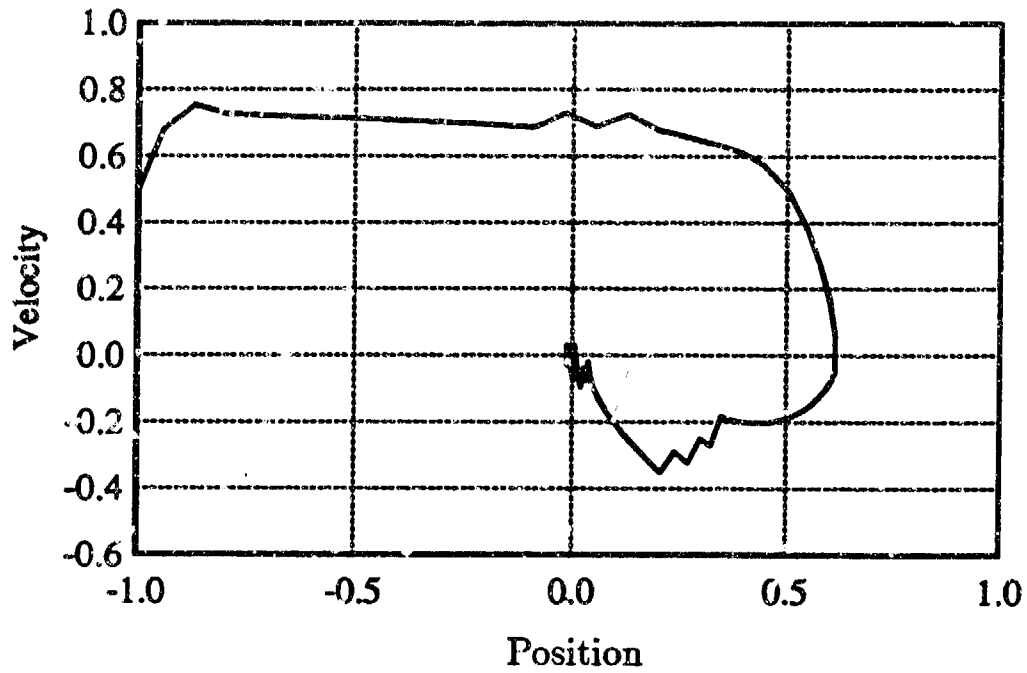


Figure 4.4. *Experiment 1*: State trajectory, $\underline{x}_0 = \{-1.0, 0.5\}$.

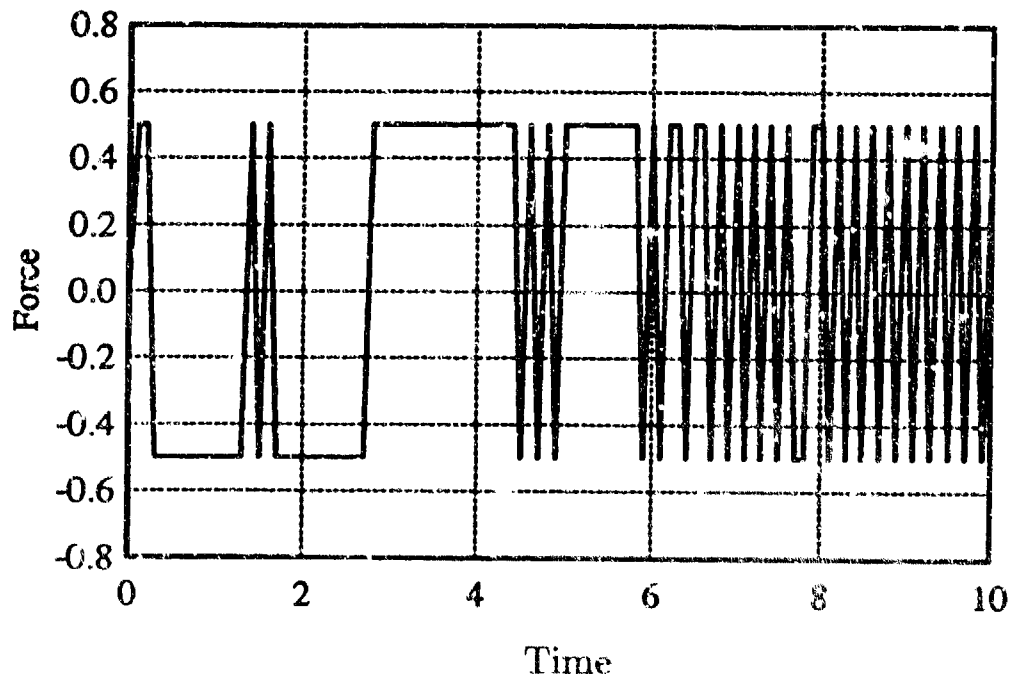


Figure 4.5. *Experiment 1*: Control history, $\underline{x}_0 = \{-1.0, 0.5\}$.

ATTACHMENT 3

4.6 Results

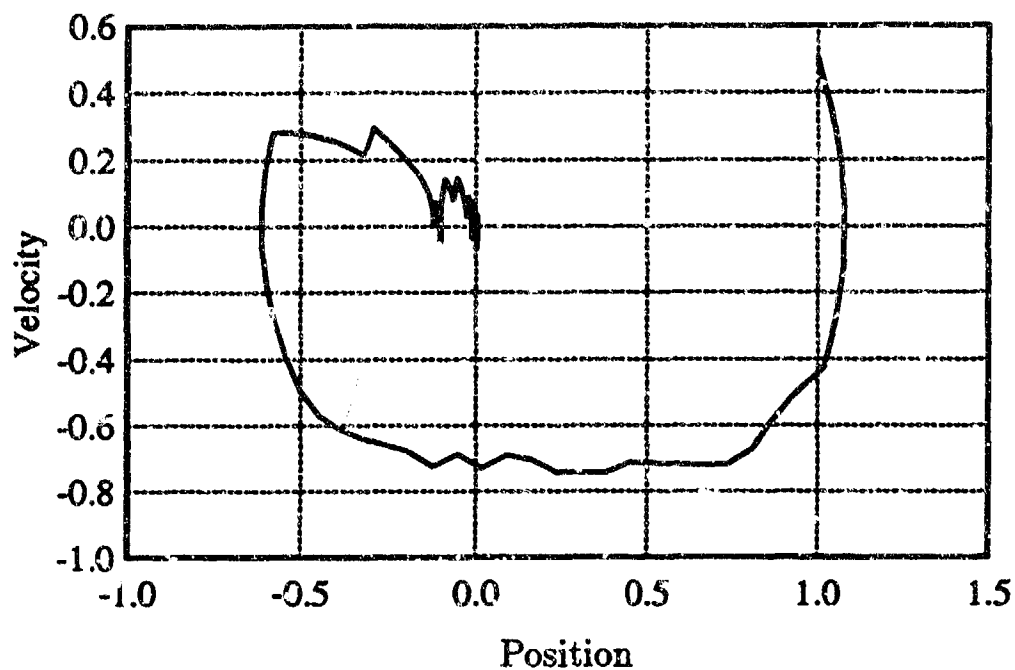


Figure 4.6. *Experiment 1*: State trajectory, $\underline{x}_0 = \{1.0, 0.5\}$.

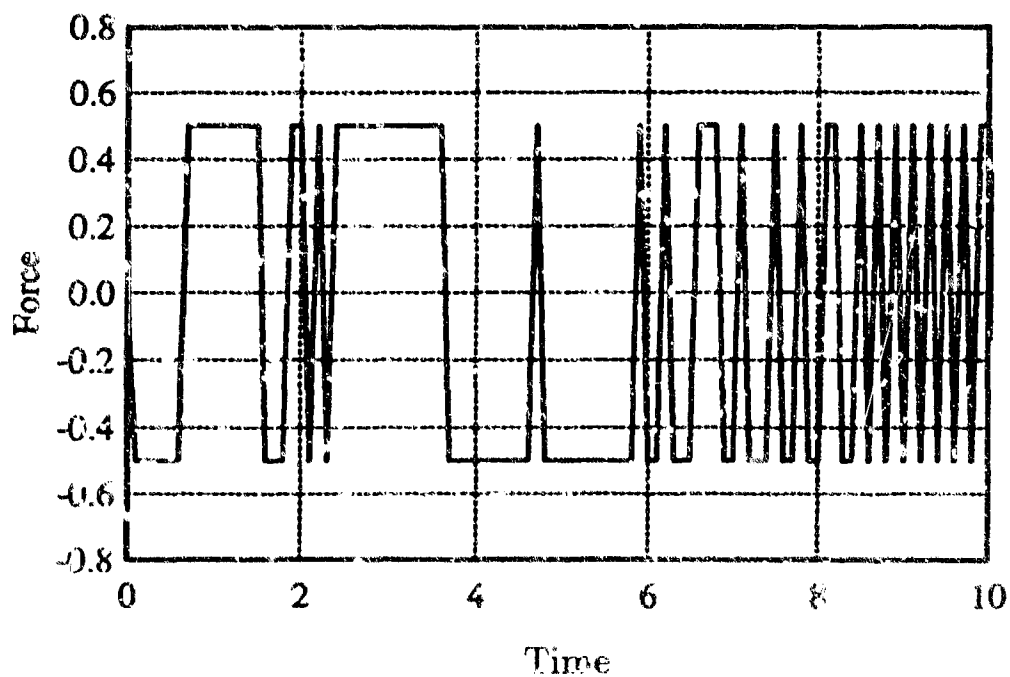


Figure 4.7. *Experiment 1*: Control history, $\underline{x}_0 = \{1.0, 0.5\}$.

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

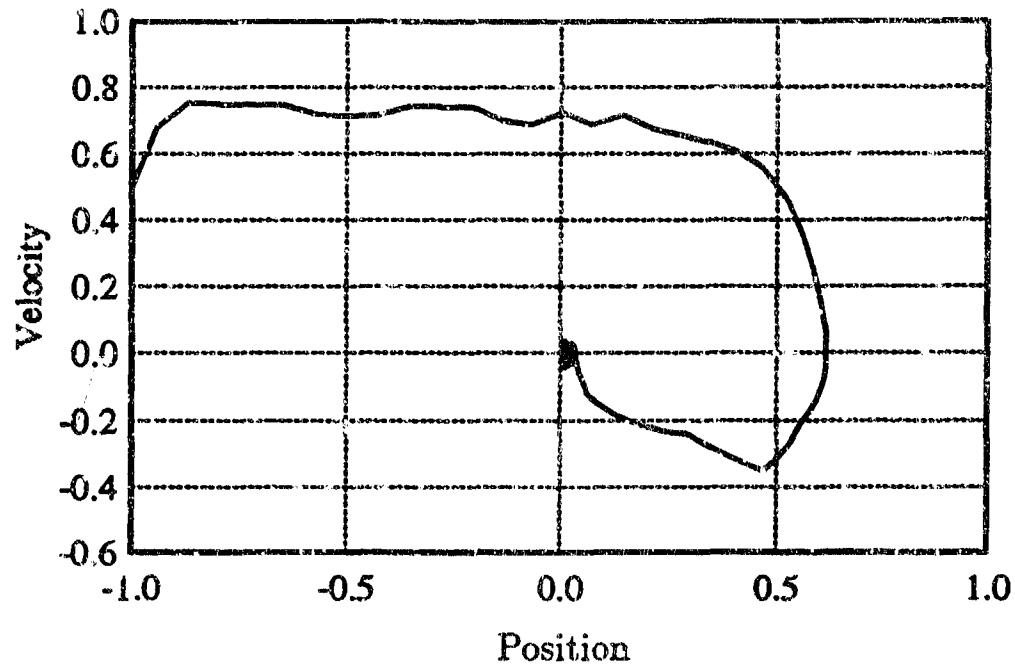


Figure 4.3. *Experiment 2*: State trajectory, $\underline{x}_0 = \{-1.0, 0.5\}$.

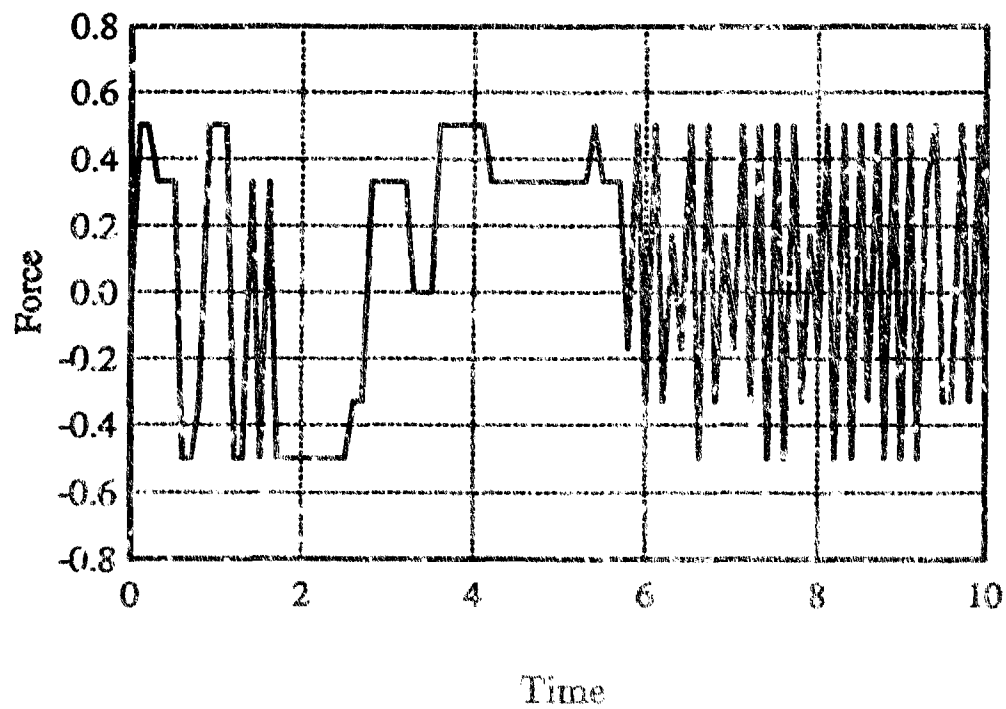


Figure 4.9. *Experiment 2*: Control history, $\underline{x}_0 = \{-1.0, 0.5\}$.

ATTACHMENT 3

4.6 Results

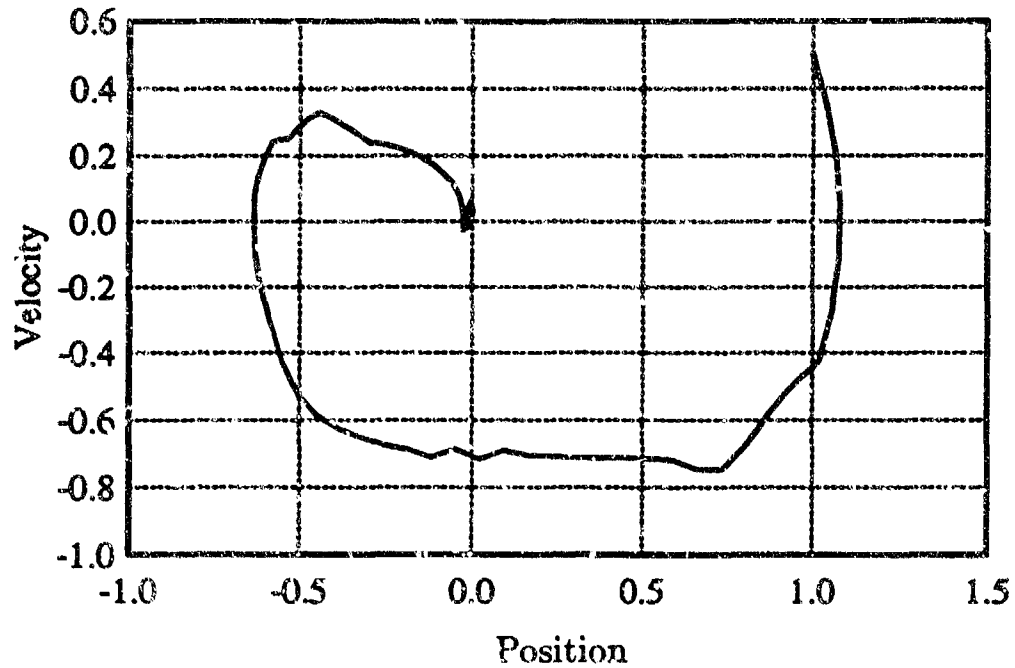


Figure 4.10. *Experiment 2*: State trajectory, $\underline{x}_0 = \{1.0, 0.5\}$.

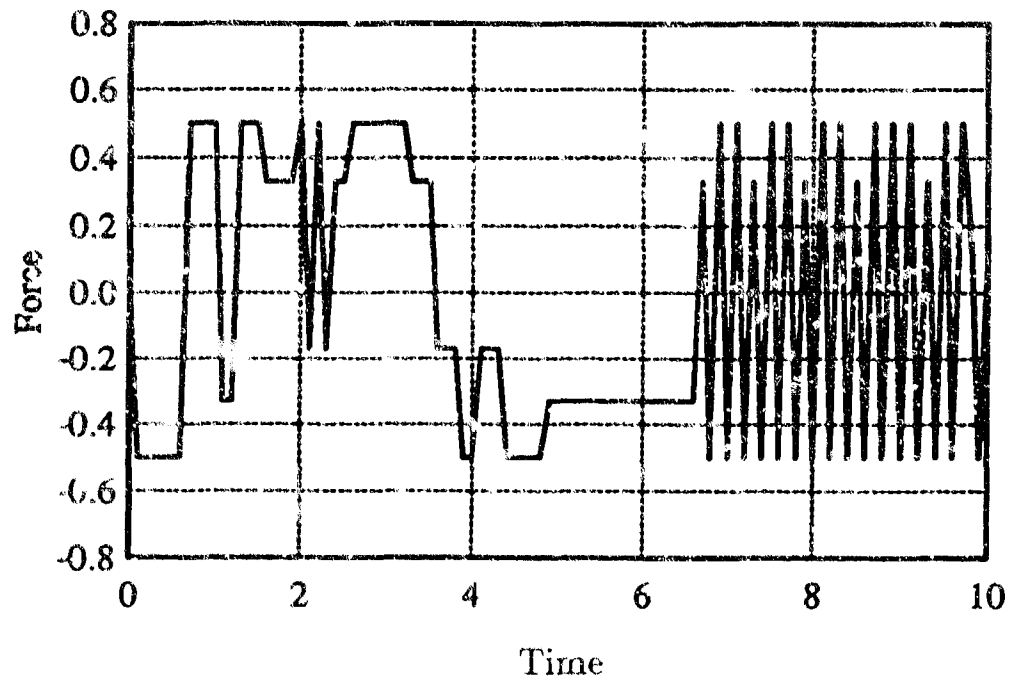


Figure 4.11. *Experiment 2*: Control history, $\underline{x}_0 = \{1.0, 0.5\}$.

ATTACHMENT 3

Chapter 4 - Policy and Value Iteration

state provided sufficient experience about performing every action in every bin. Over 2000 trials were performed in each experiment to train the learning system. If the learning rate had been a focus of the research, an explicit search procedure could have been employed. Additionally, in some experiments, the Q values did not converge to a steady state. Some of the bins were excessively large such that the optimal actions (in a continuous sense) associated with extreme points within the bin were quite different. Therefore, the Q values for such a bin, and subsequently the optimal policy, would vary as long as training continued.

All Q learning experiments learned a control policy that successfully regulated the aeroelastic oscillator. The state trajectories and control histories of the AEO, with initial conditions $\{-1.0, 0.5\}$ and $\{1.0, 0.5\}$, which resulted from the control laws learned in *Experiments 1* and *2*, appear in Figures 4.4 through 4.11. The limitation of the control to discrete levels, and the associated sharp control switching, resulted in rough state trajectories as well as limit cycles around the origin. The results illustrate the importance of a smooth control law; a continuous control law (LQR) was discussed in §2.4.2 and a characteristic state trajectory appeared in Figure 2.5. The absence of actuator dynamics and a penalty on the magnitude of the control allow the application of larger values of control to maximize reinforcement. Therefore, *Experiment 2* seldom selected a smaller or zero control force, even for bins near the origin. In *Experiment 1* the magnitude of the control was constant.

4.7 Continuous Q Learning

4.7 Continuous Q Learning

The algorithm described in §4.4 operates with both a finite set of states and discrete control actions. The optimal control a^* maximizes $Q(x, a^*)$ for the current state x . To identify the optimal control for a specific state, therefore, requires the comparison of $Q(x, a)$ for each discrete action $a \in A(x)$.² However, quantization of the input and output spaces is seldom practical or acceptable

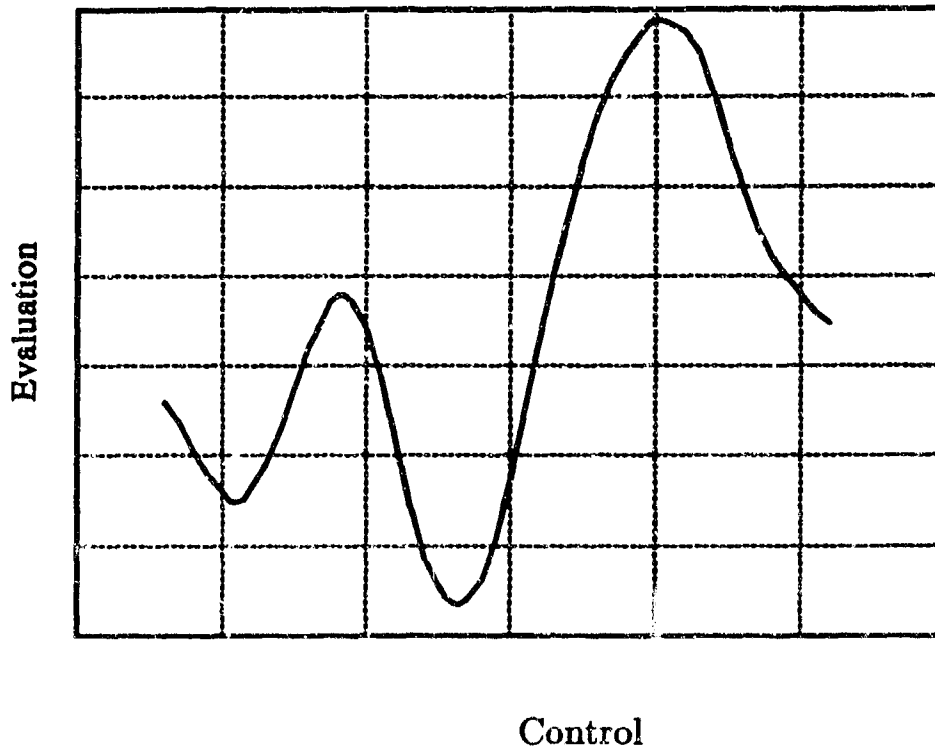


Figure 4.12. A continuous Q function for an arbitrary state x .

A potential new algorithm, related to the Q learning process of §4.4 might se-

² A finite number of Q values exist and, therefore, the maximum Q value is easily obtained.

Chapter 4 - Policy and Value Iteration

lect, for each discrete state, the optimal control action from a bounded continuum and employ a continuous Q function that maps the control levels into evaluations of the expected discounted future return (Figure 4.12). However, to identify the optimal control for a state requires the maximization of a potentially multi-modal bounded function; this extremization procedure is problematic relative to the maximization of discrete Q values. The maximization of a multi-modal function at each stage in discrete time is itself a complicated optimization problem and, although not intractable, makes any continuous Q learning procedure impractical for real-time, on-line applications. This Q learning algorithm directly extends to incorporate several control variables; the optimal controls for a state are the arguments that maximize the multidimensional Q function.

The Q learning concept may be further generalized to employ continuous inputs and continuous outputs. The algorithm maps expectations of discounted future returns as a smooth function of the state and control variables. The current state will define a hyperplane through this Q function that resembles Figure 4.12 for a single control variable. Again, a maximization of a potentially multi-modal function is required to compute each control. Although the continuous nature of the state inputs does not operationally affect the identification of an optimal control, the mapping and learning mechanisms must incorporate the local generalization of information with respect to state, a phenomenon which does not occur for discrete state bins. A continuous Q function could be represented by any function approximation scheme, such as the spatially localized connectionist network introduced in §6.

Baird [42] addressed the difficulty of determining the global maximum of a

ATTACHMENT 3

4.7 Continuous Q Learning

multi-modal function. Millington [41] proposed a direct learning control method that used a spatially localized connectionist / Analog Learning Element. The learning system defined, as a distributed function of state, a continuous probability density function for control selection.

Chapter 5

Temporal Difference Methods

Temporal difference (TD) methods comprise a class of incremental learning procedures that predict future system behavior as a function of current observations. The earliest temporal difference algorithm appeared in Samuel's checker-playing program [18].¹ Manifestations of the TD algorithm also exist in Holland's bucket brigade [13], Sutton's Adaptive Heuristic Critic [5,29], and Klopff's Drive-Reinforcement learning [12]. This chapter summarizes Sutton's unification of these algorithms into a general temporal difference theory [15] and then analyzes the similarities and distinctions between the Adaptive Heuristic Critic, Drive-Reinforcement learning, and Q learning.

¹ The phrase *temporal difference* was proposed by Sutton in 1988 [15].

Chapter 5 - Temporal Difference Methods

5.1 TD(λ) Learning Procedures

Most problems to which learning methods are applicable can be formulated as a prediction problem, where future system behavior must be estimated from transient sequences of available sensor outputs. Conventional supervised learning prediction methods associate an observation and a final outcome pair; after training, the learning system will predict the final outcome that corresponds to an input. In contrast, temporal difference methods examine temporally successive predictions of the final result to derive a similar mapping from the observations to the final outcome. Sutton demonstrates that TD methods possess two benefits relative to supervised learning prediction methods [15]. Supervised learning techniques must wait until the final outcome has been observed before performing learning calculations and, therefore, to correlate each observation with the final outcome requires storage of the sequence of observations that preceded the final result. In contrast, the TD approach avoids this storage requirement, incrementally learning as each new prediction and observation are made. This fact, and the associated temporal distribution of required calculations, make the TD algorithm amenable to running on-line with the physical plant. Through more efficient use of experience, TD algorithms converge more rapidly and to more accurate predictions. Although any learning method should converge to an equivalent evaluation with infinite experience, TD methods are guaranteed to perform better than supervised learning techniques after limited experience with a Markov decision process.

Temporal difference and conventional supervised learning are indistinguishable for single step prediction problems where the accuracy of a prediction is revealed

5.1 TD(λ) Learning Procedures

immediately. In a multi-step prediction problem, partial information pertinent to the precision of a prediction is incrementally disclosed through temporally successive observations. This second situation is more prevalent in optimal control problems. Multi-stage problems consist of several temporally sequential observations $\{x_1, x_2, \dots, x_m\}$ followed by a final result z . At each discrete time t , the learning system generates a prediction P_t of the final output, typically dependent on the current values of a weight set w . The learning mechanism is expressed as a rule for adjusting the weights.

Typically, supervised learning techniques employ a generalization of the Widrow-Hoff rule [21] to derive weight updates.²

$$\Delta w_t = \alpha(z - P_t)\Delta_w P_t \quad (5.1)$$

In contrast to (5.1), TD methods are sensitive to changes in successive predictions rather than the error between a prediction and the final outcome. Sutton has demonstrated that for a multi-step prediction problem, a TD(1) algorithm produces the same total weight changes for any observation-outcome sequence as the Widrow-Hoff procedure. The TD(1) algorithm (5.2) alters prior predictions to an equal degree.

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^t \Delta_w P_k \quad (5.2)$$

The temporal difference method generalizes from TD(1) to an algorithm that adjusts prior predictions in proportion to a factor that equals unity for the current time and

² The Widrow-Hoff rule, also known as the delta rule, requires that P_t be a linear function of w and x_t so that $\Delta_w P_t = x_t$.

ATTACHMENT 3

Chapter 5 - Temporal Difference Methods

decreases exponentially with increasing elapsed time. This algorithm is referred to as $TD(\lambda)$ and (5.3) defines the learning process, where P_{m+1} is identically z .

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^t \lambda^{t-k} \Delta_w P_k \quad (5.3)$$

$$0 \leq \lambda \leq 1 \quad (5.4)$$

An advantage of this exponential weighting is the resulting simplicity of determining future values of the summation term in (5.3).

$$\begin{aligned} S(t+1) &= \sum_{k=1}^{t+1} \lambda^{t+1-k} \Delta_w P_k = \Delta_w P_{t+1} + \sum_{k=1}^t \lambda^{t+1-k} \Delta_w P_k \\ &= \Delta_w P_{t+1} + \lambda \sum_{k=1}^t \lambda^{t-k} \Delta_w P_k = \Delta_w P_{t+1} + \lambda S(t) \end{aligned} \quad (5.5)$$

In the limiting case where $\lambda = 0$, the learning process determines the weight increment entirely by the resulting effect on the most recent prediction. This $TD(0)$ algorithm (5.6) resembles (5.1) if the final outcome z is replaced by the subsequent prediction.

$$\Delta w_t = \alpha(P_{t+1} - P_t) \Delta_w P_t \quad (5.6)$$

5.2 An Extension of $TD(\lambda)$

The TD family of learning procedures directly generalizes to accomplish the prediction of a discounted, cumulative result, such as the expected discounted future

5.3 A Comparison of Reinforcement Learning Algorithms

cost associated with an infinite horizon optimal control problem. In conformance with Sutton's notation, c_t denotes the external evaluation of the cost incurred during the time interval from $t-1$ to t . The prediction P_t , which is the output of the TD learning system, estimates the expected discounted future cost.

$$P_t \approx \sum_{n=0}^{\infty} \gamma^n c_{t+1+n} \quad (5.7)$$

The discount parameter γ specifies the time horizon with which the prediction is concerned. The recursive nature of the expression for an accurate prediction becomes apparent by rewriting (5.7).

$$P_{t-1} = c_t + \sum_{n=1}^{\infty} \gamma^n c_{t+n} = c_t + \gamma \sum_{n=0}^{\infty} \gamma^n c_{t+n+1} = c_t + \gamma P_t \quad (5.8)$$

Therefore, the error in a prediction, $(c_t + \gamma P_t) - P_{t-1}$, serves as the impetus for learning in (5.9).

$$\Delta w_t = \alpha (c_t + \gamma P_t - P_{t-1}) \sum_{k=1}^t \lambda^{t-k} \Delta_w P_k \quad (5.9)$$

5.3 A Comparison of Reinforcement Learning Algorithms

The modified TD(λ) rule (5.9) is referred to as the Adaptive Heuristic Critic (AHC) and learns to predict the summation of the discounted future values of the signal c_t . With slightly different learning equations, both Q learning and Drive-Reinforcement (DR) learning accomplish a similar objective. This section compares the form and function of these three direct learning algorithms.

ATTACHMENT 3

Chapter 5 - Temporal Difference Methods

The single step Q learning algorithm (5.10) defines a change in a Q value, which represents a prediction of expected discounted future cost, directly, rather than through adjustments to a set of weights that define the Q value.

$$Q_{t+1}(x_t, a_t) = (1 - \alpha)Q_t(x_t, a_t) + \alpha(c_t + \gamma V_t^Q(x_{t+1})) \quad (5.10)$$

Although the form of the learning equation appears different than that of the AHC or DR learning, the functionality is similar. The improved Q value $Q_{t+1}(x_t, a_t)$ equals a linear combination of the initial Q value $Q_t(x_t, a_t)$ and the sum of the cost for the current stage c_t and the discounted prediction of the subsequent discounted future cost $\gamma V_t^Q(x_{t+1})$. A similar linear combination to perform incremental improvements is achieved in both the AHC and Drive-Reinforcement learning by calculating weight changes with respect to the current weights.

Both the Drive-Reinforcement (DR) and the Adaptive Heuristic Critic learning mechanisms calculate weight changes that are proportional to the prediction error ΔP_t .

$$\Delta P_t = c_t + \gamma P_t - P_{t-1} \quad (5.11)$$

The DR learning rule is rewritten in (5.12) to conform to the current notation.

$$\Delta w_t = \Delta P_t \sum_{k=1}^{\tau} c_k f_k(\Delta x_{t-k}) \quad (5.12)$$

In the DR and AHC algorithms, a non zero prediction error causes the weights to be adjusted so that P_{t-1} would have been closer to $c_t + \gamma P_t$. The constant of

5.3 A Comparison of Reinforcement Learning Algorithms

proportionality between the weight change and the prediction error differs between DR learning and the AHC.

The Drive-Reinforcement weight changes are defined by (5.12). The limits on the summation over previous stages in time and the binary facilitation function f_s prescribe modifications to a finite number of previous predictions. An array of constants, c_k , encode a discount that determines the contribution of previous actions to the current prediction error. In contrast, the summation term in the AHC learning equation (5.9) allows all previous predictions to be adjusted in response to a current prediction error. The extent to which an old prediction is modified decreases exponentially with the elapsed time since that prediction. In the AHC algorithm, the sensitivity of the prior prediction to changes in the weights, $\Delta_w P_k$, scales the weight adjustment.

Similar to the AHC and DR learning, an incremental change in a Q value is proportional to the prediction error.

$$\Delta Q_t(x_t, a_t) = Q_{t+1}(x_t, a_t) - Q_t(x_t, a_t) = \alpha (c_t - Q_t(x_t, a_t) + \gamma V_t(x_{t+1})) \quad (5.13)$$

The expression for the prediction error in (5.13) appears different from (5.11) and warrants some explanation. $V_t(x_{t+1})$, which represents $\max_a [Q_t(x_{t+1}, a_{t+1})]$, denotes the optimal prediction of discounted future cost and, therefore, is functionally equivalent to P_t in (5.11). Moreover, the entire time basis for Q learning is shifted forward one stage with respect to the AHC or DR learning rules. As a result, Q_t operates similar to P_{t-1} in (5.11) and the symbol c_t performs the same function in (5.11) and (5.13), although the cost is measured over a different period of time in the Q learning rule than in the AHC or DR learning rules (5.9) and (5.12), respectively.

ATTACHMENT 3

Chapter 8 - Temporal Difference Methods

To summarize the comparison of direct learning algorithms, each of the three temporal difference techniques will learn a value function for the expected discounted future cost. More generally, any direct learning algorithm will maintain and incrementally improve both policy and value function information. Furthermore, although the forms of the learning equations differ slightly, each method attempts to reduce the prediction error ΔP_t .

Although the functionality of direct learning algorithms may be similar, the structure will vary. For example, Q learning distinguishes the optimal action by maximizing over the set of Q values. The Associative Control Process determines the optimal action through the biologically motivated reciprocal inhibition procedure. Furthermore, whereas Q values may equal any real number, the outputs of ACP learning centers must be non-negative, acknowledging the inability of neurons to realize negative frequencies of firing.

Chapter 6

Indirect Learning Optimal Control

Each control law derived in this chapter attempts to optimally track a reference trajectory that is generated by a linear, time-invariant reference model (6.1); optimization is performed with respect to quadratic cost functionals over finite time horizons. The notation in this chapter uses subscripts to indicate the stage in discrete time and superscripts to distinguish the plant and reference model.

$$x_{k+1}^r = \Phi^r x_k^r + \Gamma^r r_k \quad (6.1a)$$

$$y_k^r = C^r x_k^r \quad (6.1b)$$

$$y_{k+1}^p = C^p \Phi^p x_k^p + C^p \Gamma^p r_k \quad (6.1c)$$

$$y_{k+2}^p = C^p \Phi^p \Phi^p x_k^p + C^p \Phi^p \Gamma^p r_k + C^p \Gamma^p r_{k+1} \quad (6.1d)$$

Although a few subsequent values of the command input after r_k may be characterized at time k , the future input sequence will be largely unknown. To apply

Chapter 6 - Indirect Learning Optimal Control

infinite horizon, linear quadratic (LQ) control techniques to the tracking problem requires a description of the future command inputs. Furthermore, in a multi-objective mission, such as aircraft flight involving several different flight conditions, the existence of future maneuvers should negligibly influence the optimization of performance during the current operation. Finally, optimizations over unnecessarily long time frames may require prohibitively long computations. Therefore, finite horizon LQ control directly addresses relevant control problems.

6.1 Single-Stage Quadratic Optimization

The control objective is to minimize the quadratic cost functional J_k which penalizes the current control expenditure and the output error e_{k+1} , given by the difference between the reference output and the system output at time $k+1$. The weighting matrices R and Q are symmetric and positive definite.

$$J_k = \frac{1}{2} [e_{k+1}^T Q e_{k+1} + u_k^T R u_k] \quad (6.2)$$

$$e_{k+1} = y_{k+1}^r - y_{k+1} \quad (6.3)$$

A single first-order necessary condition defines the condition for a control u_k to minimize the cost functional J_k [22,23].

$$\frac{\partial J_k}{\partial u_k} = 0 \quad (6.4a)$$

6.1 Single-Stage Quadratic Optimization

$$\left(\frac{\partial e_{k+1}}{\partial u_k} \right)^T Q e_{k+1} + R u_k = 0 \quad (6.4b)$$

6.1.1 Linear Compensation

Assuming a minimum-phase plant, the linear compensator is the solution to the problem of optimal tracking, with respect to the cost functional (6.2), of the reference system (6.1) with a linear, time-invariant system (6.5). Applied to a nonlinear system, this control law serves as a baseline with which to compare a single-stage, indirect learning control law. The fact that indirect learning control is a model based technique distinguishes the approach from direct learning control algorithms.

$$x_{k+1} = \Phi x_k + \Gamma u_k \quad (6.5a)$$

$$y_k = C x_k \quad (6.5b)$$

$$y_{k+1} = C \Phi x_k + C \Gamma u_k \quad (6.5c)$$

That the partial derivative of e_{k+1} with respect to u_k is independent of u_k implies that (6.4b) is linear in the optimal control. Therefore, (6.4b) may be written as an exact analytic expression for the optimal control [24].

$$\frac{\partial e_{k+1}}{\partial u_k} = -C\Gamma \quad (6.6)$$

$$u_k = [(C\Gamma)^T Q C \Gamma + R]^{-1} (C\Gamma)^T Q [C^T \Phi^T x_k^* + C^T \Gamma^T r_k - C \Phi x_k] \quad (6.7)$$

ATTACHMENT 3

Chapter 6 - Indirect Learning Optimal Control

The sufficient condition for this control to be a minimizing solution, that $\frac{\partial^2 J_k}{\partial u_k^2}$ is non-negative, is satisfied for physical systems.

$$\frac{\partial J_k}{\partial u_k} = -(C\Gamma)^T Q e_{k+1} + R u_k = 0 \quad (6.8)$$

$$\frac{\partial^2 J_k}{\partial u_k^2} = (C\Gamma)^T Q (C\Gamma) + R \geq 0 \quad (6.9)$$

6.1.2 Learning Control

In contrast to the single-stage linear compensator, the single-stage, indirect learning controller is the solution to the problem of optimal tracking of the reference system (6.1) by a nonlinear, time-invariant system (6.10), with respect to the cost functional (6.2). Again, the zero dynamics of the plant must be stable. The expression for the discrete time state propagation (6.10a) includes the a priori linear terms from (6.5) as well as two nonlinear terms: $f_k(x_k, u_k)$ represents the initially unmodeled dynamics that have been learned by the system, and $\Psi_k(x_k)$ represents any state dependent dynamics not captured by either the a priori description or the learning augmentation. The assumption of an absence of time varying disturbances and noise from the real system implies that all dynamics are spatially dependent and will be represented in the model. The system outputs are a known linear combination of the states. The notation explicitly shows the time dependence of f_k and Ψ_k , which change as learning progresses; f_k will acquire more of the subtleties in the dynamics and, consequently, Ψ_k will approach zero.

$$x_{k+1} = \Phi x_k + \Gamma u_k + f_k(x_k, u_k) + \Psi_k(x_k) \quad (6.10a)$$

6.1 Single-Stage Quadratic Optimization

$$y_k = Cx_k \quad (6.10b)$$

$$y_{k+1} = C\Phi x_k + C\Gamma u_k + C f_k(x_k, u_k) + C\Psi_k(x_k) \quad (6.10c)$$

In this formulation, the first-order necessary condition (6.4) for a control u_k to be optimal with respect to (6.2) cannot be directly solved for u_k because of the presence of the term $f_k(x_k, u_k)$ which may be nonlinear in u_k . The Newton-Raphson iterative technique [25] addresses this nonlinear programming problem by linearizing $f_k(x_k, u_k)$ with respect to u at u_{k-1} . $\frac{\partial f}{\partial u}$ is the Jacobian matrix of f_k with respect to u , evaluated at $\{x_k, u_{k-1}\}$. Using this approximation for $f_k(x_k, u_k)$, y_{k+1} assumes a form linear in u_k and (6.4) may be written as an analytic expression for u_k in terms of known quantities.

$$f_k(x_k, u_k) \approx f_k(x_k, u_{k-1}) + \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} (u_k - u_{k-1}) \quad (6.11)$$

$$y_{k+1} \approx C\Phi x_k + C\Gamma u_k + C f_k(x_k, u_{k-1}) + C \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} (u_k - u_{k-1}) + C\Psi_k(x_k) \quad (6.12)$$

$$\frac{\partial e_{k+1}}{\partial u_k} = -C\Gamma - C \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} \quad (6.13)$$

The solution (6.14) is the first Newton-Raphson estimate for the optimal controls; a pseudo-inverse may be used in (6.14) if the full matrix inversion does not exist. Subsequent estimates u_k^i may be derived by linearizing (6.10) about u_k^{i-1} . However, the estimate obtained in the first iteration is often sufficiently accurate because the initial linearization is about u_{k-1} and the admissible change in control $\Delta u_k = u_k - u_{k-1}$ will be limited by actuator rate limits and a sufficiently small discrete time interval [25,26].

Chapter 6 - Indirect Learning Optimal Control

The form of the learning augmented control law closely resembles the linear control law (6.7). In (6.14) CF is modified by $\frac{\partial f}{\partial u}$ which describes the variation in control effectiveness that was unmodeled in the linear system. The final three terms of (6.14) are not present in (6.7) and enter from the nonlinear terms in (6.10).

$$u_k = \left[\left(CI + C \frac{\partial f}{\partial u} \right)^T Q \left(CI + C \frac{\partial f}{\partial u} \right) + R \right]^{-1} \left[\left(CI + C \frac{\partial f}{\partial u} \right)^T Q \left[C^T \Phi^T x_k + C^T I - C \Phi x_k - C f_k(x_k, u_{k-1}) + \frac{\partial f}{\partial x} \right] + C^T u_{k-1} \right] \quad (6.14)$$

A simple adaptive estimate of the unmodeled dynamics at a time k is generated by solving (6.10) at the previous time step for Ψ_{k-1} and assuming $\Psi_k(x_k) \approx \Psi_{k-1}(x_{k-1})$. This adaptive technique is susceptible to unmodeled disturbances present in the real system [27]

$$\hat{\Psi}_k(x_k) = \hat{\Psi}_{k-1}(x_{k-1}) - I'u_{k-1} - f_k(x_{k-1}, u_{k-1}) \quad (6.15)$$

6.2 Penalizing Control Rate

In parallel to the elements may be used to derive a control law that is optimal with respect to a cost functional that also penalizes changes in control. The Δu_k , Δu_k components in (6.16) discourages large control rates that may not be achievable, as a result of physical limitations of the actuators. The control law

6.2 Two-Step Quadratic Optimization

(6.18) resembles (6.14) with the addition of two terms involving S , which is symmetric and positive definite.

$$J_k = \frac{1}{2} \left[e_{k+1}^T Q e_{k+1} + u_k^T R u_k + \Delta u_k^T S \Delta u_k \right] \quad (6.16)$$

$$\Delta u_k = u_k - u_{k-1} \quad (6.17)$$

$$\begin{aligned} u_k = & \left[\left(C\Gamma + C \frac{\partial f}{\partial u} \right)^T Q \left(C\Gamma + C \frac{\partial f}{\partial u} \right) + R + S \right]^{-1} \\ & \left[\left(C\Gamma + C \frac{\partial f}{\partial u} \right)^T Q \left(C^r \Phi^r x_k^r + C^r \Gamma^r r_k - C \Phi x_k \right. \right. \\ & \left. \left. - C f_k(x_k, u_{k-1}) + C \frac{\partial f}{\partial u} u_{k-1} - C \Psi_k(x_k) \right) + S u_{k-1} \right] \end{aligned} \quad (6.18)$$

6.2 Two-Step Quadratic Optimization

This section parallels the discussion of §6.1 to derive the control laws that are optimal with respect to a two time step, quadratic cost functional (6.19); a few new issues arise. The expression for the reference output two time steps into the future (6.1d) involves a future value of the command input r_{k+1} . This derivation assumes that $r_{k+1} \approx r_k$.

$$J_k = \frac{1}{2} \left[e_{k+1}^T Q^1 e_{k+1} + e_{k+2}^T Q^2 e_{k+2} + u_k^T R^0 u_k + u_{k+1}^T R^1 u_{k+1} \right] \quad (6.19)$$

Chapter 6 - Indirect Learning Optimal Control

Two necessary conditions are required to define a control which is optimal with respect to (6.19). Each of the weighting matrices in (6.19) is symmetric and positive definite.

$$\frac{\partial J_k}{\partial u_k} = 0 \quad (6.20a)$$

$$\left(\frac{\partial e_{k+1}}{\partial u_k} \right)^T Q^1 e_{k+1} + \left(\frac{\partial e_{k+2}}{\partial u_k} \right)^T Q^2 e_{k+2} + R^0 u_k = 0 \quad (6.20b)$$

$$\frac{\partial J_k}{\partial u_{k+1}} = 0 \quad (6.21a)$$

$$\left(\frac{\partial e_{k+1}}{\partial u_{k+1}} \right)^T Q^1 e_{k+1} + \left(\frac{\partial e_{k+2}}{\partial u_{k+1}} \right)^T Q^2 e_{k+2} + R^1 u_{k+1} = 0 \quad (6.21b)$$

6.2.1 Linear Compensation

The output of the linear system (6.5) two time steps into the future is easily determined because the dynamics are assumed to be entirely known.

$$y_{k+2} = C\Phi\Phi x_k + C\Phi\Gamma u_k + C\Gamma u_{k+1} \quad (6.22)$$

$$e_{k+2} = y_{k+2}^r - y_{k+2} \quad (6.23)$$

The simultaneous solution of (6.20b) and (6.21b) yields a solution for u_k ; an expression for u_{k+1} , calculated at time k , is also available. However, to parallel the learning control process, this control will be recalculated at the next time step. To

6.2 Two-Step Quadratic Optimization

illustrate the similarity of the linear and learning control laws, as well as to express the laws in a compact form, several substitutions are defined.

$$A = C\Gamma \quad (6.24a)$$

$$B = C\Phi\Gamma \quad (6.24b)$$

$$\Lambda = A^T Q^2 A + R^1 \quad (6.25a)$$

$$\Upsilon = B^T Q^2 A \quad (6.25b)$$

$$\Theta = A^T Q^1 \quad (6.25c)$$

$$\Xi = (B^T - \Upsilon \Lambda^{-1} A^T) Q^2 \quad (6.25d)$$

$$\begin{aligned} u_k = & \left[A^T Q^1 A + B^T Q^2 B + R^0 - \Upsilon \Lambda^{-1} \Upsilon^T \right]^{-1} \\ & \left[(\Theta C^T \Phi^T + \Xi C^T \Phi^T \Phi^T) x_k^T \right. \\ & + (\Theta C^T \Gamma^T + \Xi (C^T \Phi^T \Gamma^T + C^T \Gamma^T)) r_k \\ & \left. - (\Theta C \Phi + \Xi C \Phi \Phi) z_k \right] \end{aligned} \quad (6.26)$$

6.2.2 Learning Control

For the nonlinear system, the output y_{k+2} (6.27a) must be approximated by known quantities that are linear in u_k and u_{k+1} . First, the nonlinear terms in (6.27a) are evaluated at the current time k rather than at $k+1$ and an approximation x_{k+1} is derived for the next state. Additionally, the learned dynamics are

ATTACHMENT 3

Chapter 6 - Indirect Learning Optimal Control

estimated by linearizing $f_k(x_{k+1}, u_{k+1})$ about the point $\{x_k, u_{k-1}\}$.

$$y_{k+2} = C\Phi x_{k+1} + C\Gamma u_{k+1} + Cf_{k+1}(x_{k+1}, u_{k+1}) + C\Psi_{k+1}(x_{k+1}) \quad (6.27a)$$

$$\approx C\Phi \hat{x}_{k+1} + C\Gamma u_{k+1} + Cf_k(x_{k+1}, u_{k+1}) + C\Psi_k(\hat{x}_{k+1}) \quad (6.27b)$$

$$\hat{x}_{k+1} = \Phi x_k + \Gamma u_k + f_k(x_k, u_{k-1}) + \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} (u_k - u_{k-1}) + \Psi_k(x_k) \quad (6.28)$$

$$f_k(x_{k+1}, u_{k+1}) \approx f_k(x_k, u_{k-1}) + \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} (u_{k+1} - u_{k-1}) + \left. \frac{\partial f}{\partial x} \right|_{x_k, u_{k-1}} (\hat{x}_{k+1} - x_k) \quad (6.29)$$

$$\begin{aligned} y_{k+2} \approx & C\Phi \hat{x}_{k+1} + C\Gamma u_{k+1} + Cf_k(x_k, u_{k-1}) + C \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} (u_{k+1} - u_{k-1}) \\ & + C \left. \frac{\partial f}{\partial x} \right|_{x_k, u_{k-1}} (\hat{x}_{k+1} - x_k) + C\Psi_k(\hat{x}_{k+1}) \end{aligned} \quad (6.30)$$

Using this approximation for y_{k+2} , the simultaneous solution of (6.20b) and (6.21b) yields an expression for u_k in terms of (6.25) and (6.31). The variables A and B include both the linear components of the a priori model as well as learned state dependent corrections. $\frac{\partial f}{\partial u}$ is a correction to the constant Γ matrix and $\frac{\partial f}{\partial x}$ is a correction to the constant Φ matrix.

$$A = C\Gamma + C \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} \quad (6.31a)$$

$$B = C\Phi\Gamma + C\Phi \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} + C \left. \frac{\partial f}{\partial x} \right|_{x_k, u_{k-1}} \Gamma + C \left. \frac{\partial f}{\partial u} \right|_{x_k, u_{k-1}} \left. \frac{\partial f}{\partial x} \right|_{x_k, u_{k-1}} \quad (6.31b)$$

Although the simultaneous solution of the first-order necessary conditions also yields an expression for u_{k+1} at $k = u_k$ is calculated on every time step. This control

6.3 Implementation and Results

law resembles the form of (6.26) and introduces several terms associated with the nonlinear dynamics.

$$\begin{aligned}
 u_k = & \left[A^T Q^1 A + B^T Q^1 B + R^0 - \Upsilon \Lambda^{-1} \Upsilon^T \right]^{-1} \\
 & \left[(\Theta C^T \Phi^r + \Xi C^T \Phi^r \Phi^r) x_k^r \right. \\
 & + (\Theta C^T \Gamma^r + \Xi (C^T \Phi^r \Gamma^r + C^T \Gamma^r)) r_k \\
 & - \left(\Theta C \Phi + \Xi \left(C \Phi \Phi + C \frac{\partial f}{\partial x} \Phi - C \frac{\partial f}{\partial x} \right) \right) x_k \\
 & + \left(\Theta C \frac{\partial f}{\partial u} + \Xi \left(C \Phi \frac{\partial f}{\partial u} + C \frac{\partial f}{\partial u} + C \frac{\partial f}{\partial x} \frac{\partial f}{\partial u} \right) \right) u_{k-1} \\
 & - \left(\Theta C + \Xi \left(C \Phi + C + C \frac{\partial f}{\partial x} \right) \right) f_k(x_k, u_{k-1}) \\
 & - \left(\Theta C + \Xi \left(C \Phi + C \frac{\partial f}{\partial x} \right) \right) \Psi_k(x_k) \\
 & \left. - \Xi C \Psi_k(\hat{x}_{k+1}) \right] \quad (6.32)
 \end{aligned}$$

6.2.3 Multi-stage Quadratic Optimization

The arguments presented in §6.1 and thus far in §6.2 may be generalized to derive a control law which is optimal with respect to a cost functional (6.32) that looks n time steps into the future. The solution of this problem, however, will require assumptions about the propagation of the command input r for n future time steps. Additionally, the algebra required to write an explicit expression for u_k becomes involved and the necessary linearizations become less accurate.

$$J_k = \frac{1}{2} \sum_{i=1}^n \left[c_{k+i}^T Q^1 c_{k+i} + u_{k+i-1}^T R^1 u_{k+i-1} + \Lambda u_{k+i-1}^T S^1 \Lambda u_{k+i-1} \right] \quad (6.33)$$

Chapter 6 - Indirect Learning Optimal Control

6.3 Implementation and Results

6.3.1 Reference Model

The reference model, which generates trajectories that the plant states attempt to follow, exhibits a substantial influence on the closed-loop system performance. While a reference model that does not satisfy performance specifications will yield an unacceptable closed-loop system, a reference model that demands unrealistic (i.e. unachievable) state trajectories may introduce instability through control saturation. Therefore, the reference model must be selected to yield satisfactory performance given the limitations of the dynamics [28].

The reference model was selected to be the closed-loop system that resulted from applying the optimal control from a linear quadratic design, to the aeroelastic oscillator dynamics linearized at the origin [29]. The discrete time representation of the reference model as well as the AEO linear dynamics are presented for $\Delta t = 0.1$.

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (6.34a)$$

$$R = 1.0 \times 10^{-7} \quad (6.34b)$$

$$C = C^T = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \quad (6.35)$$

$$\Phi = \begin{bmatrix} 0.994798 & 0.106070 \\ -0.106070 & 1.122083 \end{bmatrix} \quad (6.36a)$$

$$\Gamma = \begin{bmatrix} 0.005202 \\ 0.106070 \end{bmatrix} \quad (6.36b)$$

6.3 Implementation and Results

$$\Phi^r = \begin{bmatrix} 0.905124 & 0.000286 \\ -0.905124 & -0.000286 \end{bmatrix} \quad (6.37a)$$

$$\Gamma^r = \begin{bmatrix} 0.094876 \\ 0.905124 \end{bmatrix} \quad (6.37b)$$

Design of an optimal control law might be accomplished with a learning system that incrementally increases closed-loop performance requirements, by adjusting the reference trajectory in regions of the state space where the current control law can achieve near perfect tracking. This is a topic for future research.

6.3.2 Function Approximation

The discussion of direct learning optimal control (§3 – §5) focused on learning system architectures and algorithms which, themselves, operate as controllers. The discussion of indirect learning optimal control is primarily concerned with the manner in which exponential information about unmodeled dynamics may be incorporated into optimal control laws. The method by which a supervised learning system approximates the initially unmodeled dynamics is a separate issue which has received much investigation [21,30,31,32].

After a brief summary, this thesis abstracts the technique for realizing the nonlinear mapping $f(x, u)$ into a *black box* which provides the desired information: $f_k(x_k, u_{k-1})$, $\frac{\partial f_k}{\partial u}|_{x_k, u_{k-1}}$, and $f_k(x_{k-1}, u_{k-1})$.

A spatially localized connectionist network is used to represent the mapping from the space of states and control to the initially unmodeled dynamics. The linear-Gaussian network achieves spatial locality by coupling a local basis function with an influence function [4,28]. The influence function, which determines the region in

ATTACHMENT 3

Chapter 6 - Indirect Learning Optimal Control

the input space of applicability of the associated basis function, is a hyper-Gaussian; the basis function is a hyperplane.

The contribution of a basis function to the network output equals the product of the basis function and the influence function, evaluated at the current input, where the influence function is normalized so that the sum of all the influence functions at the current input is unity [28]. The control law provides to the network an estimate of the model errors, $x_k - \Phi x_{k-1} - \Gamma u_{k-1}$. The supervised learning procedure follows an incremental gradient descent algorithm in the space of the network errors by adjusting the parameters that describe the slopes and offsets of the basis functions.

In terms of equations and for arbitrary input and output dimensions, $Y(x)$ is the network output evaluated at the current input x . The network consists of n nodes (i.e. basis function and influence function pairs).

$$Y(x) = \sum_{i=1}^n L_i(x) \Gamma_i(x) \quad (6.38a)$$

$L_i(x)$ is the evaluation of the i^{th} basis function at the current input. W_i is a weight matrix that defines the slopes of the hyperplane and b_i is a bias vector. x_0 defines the center of the influence function.

$$L_i(x) = W_i(x - x_0) + b_i \quad (6.38b)$$

$\Gamma_i(x)$ is the i^{th} normalized influence function and is not related to the discrete time B matrix. $G_i(x)$ is the i^{th} influence function evaluated at x , where the diagonal

ATTACHMENT 3

6.3 Implementation and Results

matrix D_i represents the spatial decays of the Gaussians.

$$\Gamma_i(v) = \frac{G_i(x)}{\sum_{j=1}^n G_j(x)} \quad (6.38c)$$

$$G_i(x) = \exp \left[-\frac{1}{2} (x - x_0)^T D_i^2 (x - x_0) \right] \quad (6.38d)$$

The learning network had three inputs (position, velocity, and control) and two outputs (unmodeled position and velocity dynamics). in addition, the partial derivatives of the system outputs with respect to the inputs were available.

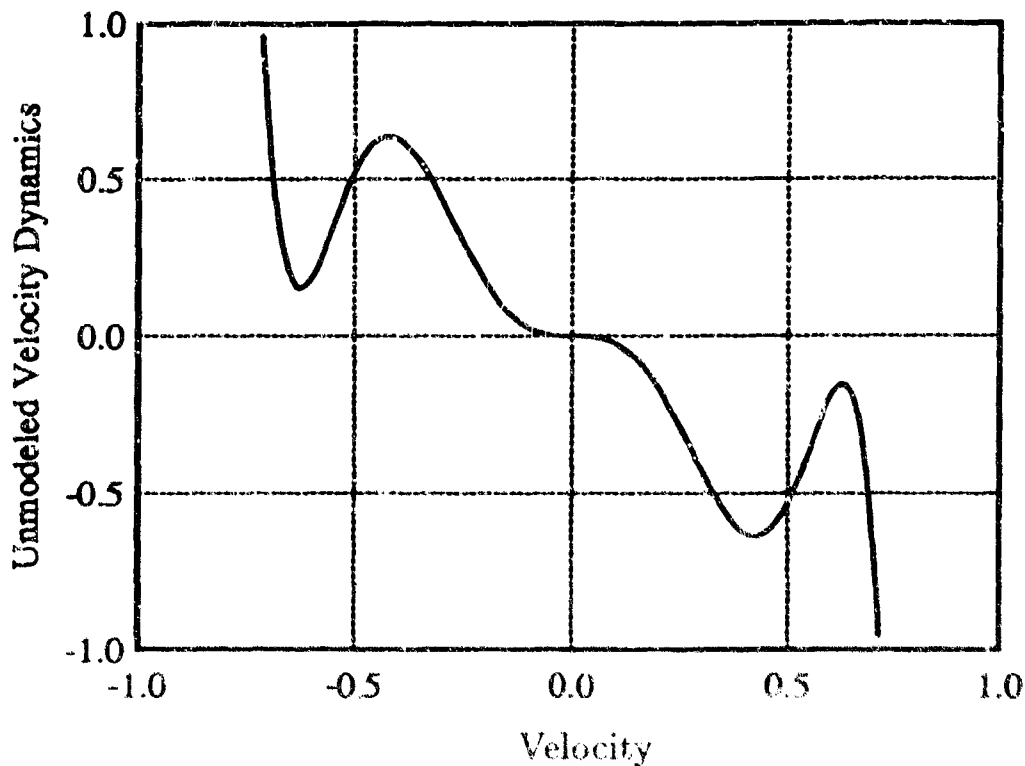


Figure 6.1. The initially unmodeled velocity dynamics $g(x_2)$ as a function of velocity x_2 .

Chapter 6 - Indirect Learning Optimal Control

The AEO dynamics are repeated in (6.39). The learning system must synthesize on-line the initial model uncertainty, which consists of the nonlinear term $g(x_2)$ in the velocity dynamics.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & nA_1U - 2\beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} F + \begin{bmatrix} 0 \\ g(x_2) \end{bmatrix} \quad (6.39a)$$

$$g(x_2) = \frac{1}{1000} \left[-\frac{nA_3}{U}(1000x_2)^3 + \frac{nA_5}{U^3}(1000x_2)^5 - \frac{nA_7}{U^5}(1000x_2)^7 \right] \quad (6.39b)$$

The manner in which the position and control enter the dynamics is linear and perfectly modeled. Therefore, the function f will be independent of the position and control (Figure 6.1). Additional model errors may be introduced to the a priori model by altering the coefficients that describe how the state and control enter the linear dynamics. The learning system will approximate all model uncertainty.

Although the magnitude of the control had been limited in the direct learning control results, where the reinforcement signal was only a function of the state error, limitation of the control magnitude was not necessary for indirect learning controllers because control directly entered the cost functional.

6.3.3 Single-Stage Quadratic Optimization Results

For the minimization of the weighted sum of the squares of the current control and the succeeding output error, the performance of the learning enhanced control law (6.14) was compared to the associated linear control law (6.7), in the context of the aeroelastic oscillator. Results appear in Figures 6.2 through 6.9. The control and reference model were updated at 10Hz; the AEO simulation was integrated using a fourth-order Runge-Kutta algorithm with a step size of 0.005.

6.3 Implementation and Results

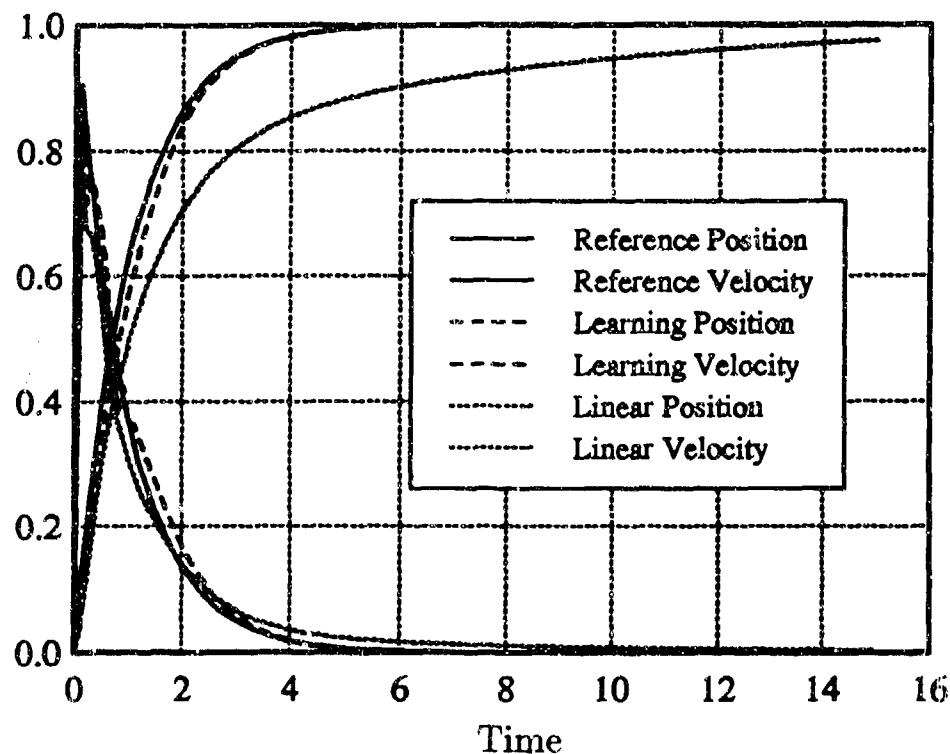


Figure 6.2. Position and velocity time histories for the reference model as well as the AEO controlled by the linear and learning control laws, for the command $r = 1$ and the initial condition $\underline{x}_0 = \{0, 0\}$.

Figure 6.2 illustrates the reference model position and velocity time histories as well as two sets of state time histories for the AEO controlled by the linear and learning control laws. The presence of unmodeled nonlinear dynamics prevented the linear control law from closely tracking the reference position. In contrast, the learning system closely followed the reference, after sufficient training. Both control laws maintained the velocity near the reference. Although the full learning control law (6.14) was implemented to produce these results, knowledge of the form of the AEO nonlinearities could have been used to eliminate the terms containing $\frac{\partial f}{\partial u}$. Figure 6.3 represents the errors between the AEO states and the reference

ATTACHMENT 3

Chapter 6 - Indirect Learning Optimal Control

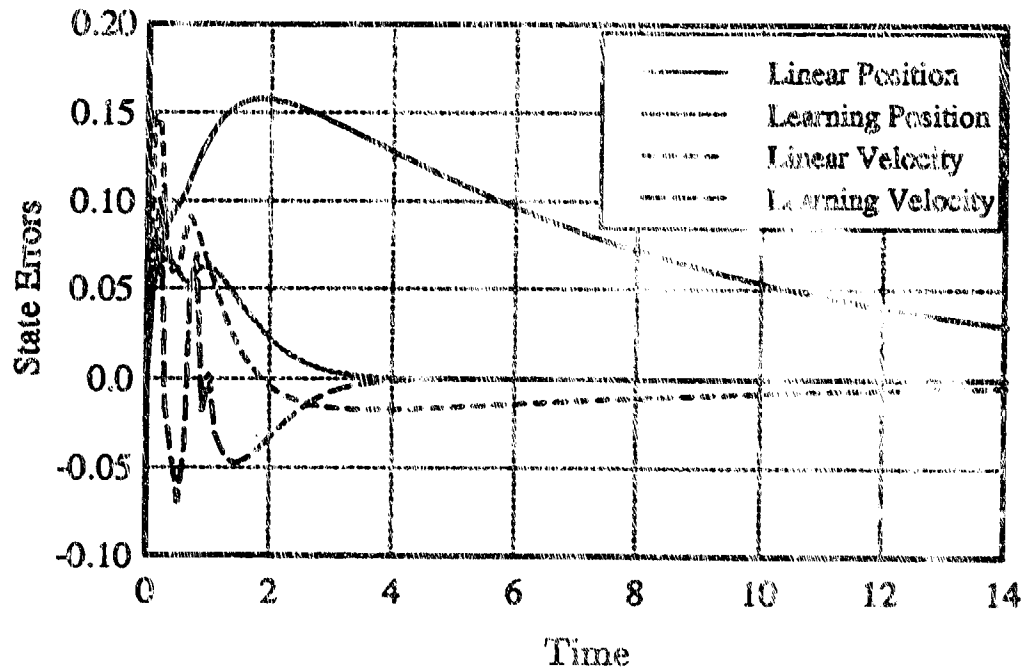


Figure 6.3. The state errors for the AEO controlled by the linear and learning control laws.

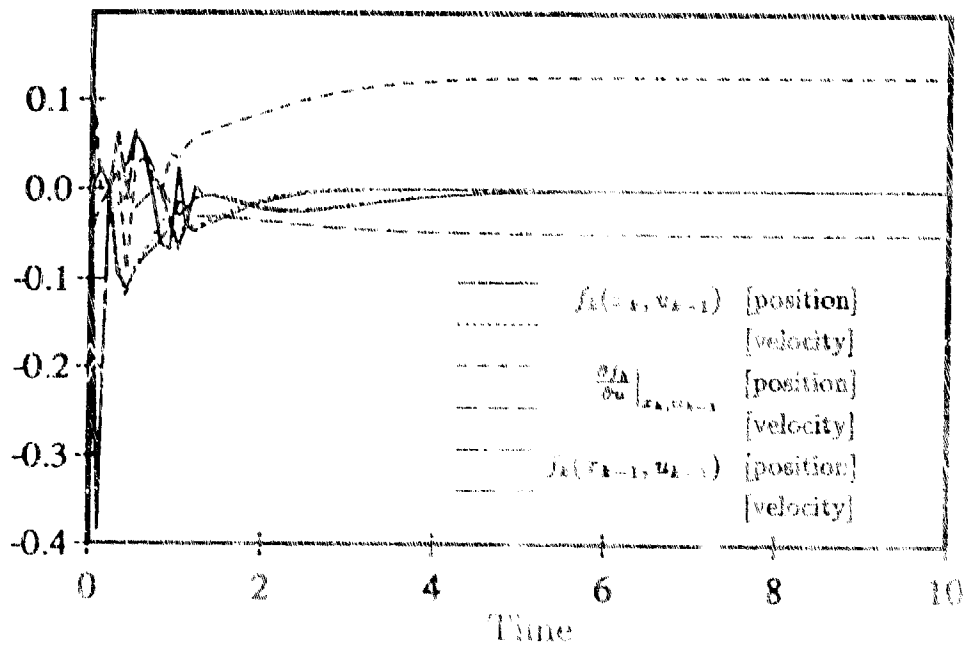


Figure 6.4. The network outputs that were used to compute u_k for the learning control law.

ATTACHMENT 3

6.3 Implementation and Results

model, for both control law designs. In a separate experiment that introduced model uncertainty in the linear dynamics, the linear control law (6.7) failed to track a command step input with zero steady-state error. The learning control law results looked similar to Figure 6.2.

The specifics of the incremental function approximation were not a focus of this indirect learning control research. The learning process involved numerous trials (more than 1000) from random initial states within the range $\{-1.0, 1.0\}$; the commanded position was also selected randomly between the same limits. The allocation of network resources (i.e. adjustable parameters) and the selection of learning rates involved heuristics. Moreover, the learning performance depended strongly on these decisions. Automation of the network design process would have greatly facilitated this research.

The learning control law requires the values of the network outputs at the current state and the previous control, as well as the partial derivative of the network outputs with respect to the control, at the current state and the previous control. Additionally, the adaptive term $\Psi_k(x_k)$ requires the values of the network outputs at the previous state and the previous control. The network outputs, which appear in Figure 6.4, change most rapidly when the velocity is not near zero, i.e. at the beginning of a trial. Some rapid changes in the network outputs resulted from learning errors where f did not accurately approximate the nonlinear dynamics.

For the learning control law, the control as well as the terms of (6.14) that comprise the control appear in Figure 6.5. The control for the linear control law and the individual terms of (6.7) appear in Figure 6.6.

After substantial training, some errors remained in the network's approxima-

ATTACHMENT 3

Chapter 6: Indirect Learning Optimal Control

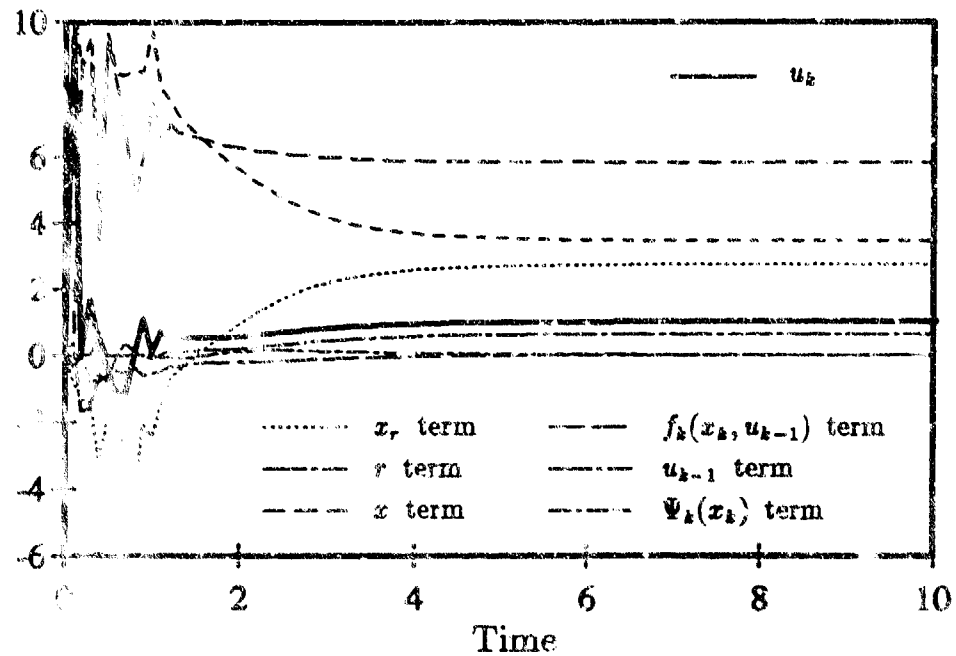


Figure 6.6. The control u_k and the constituent terms of the learning control law (6.14).

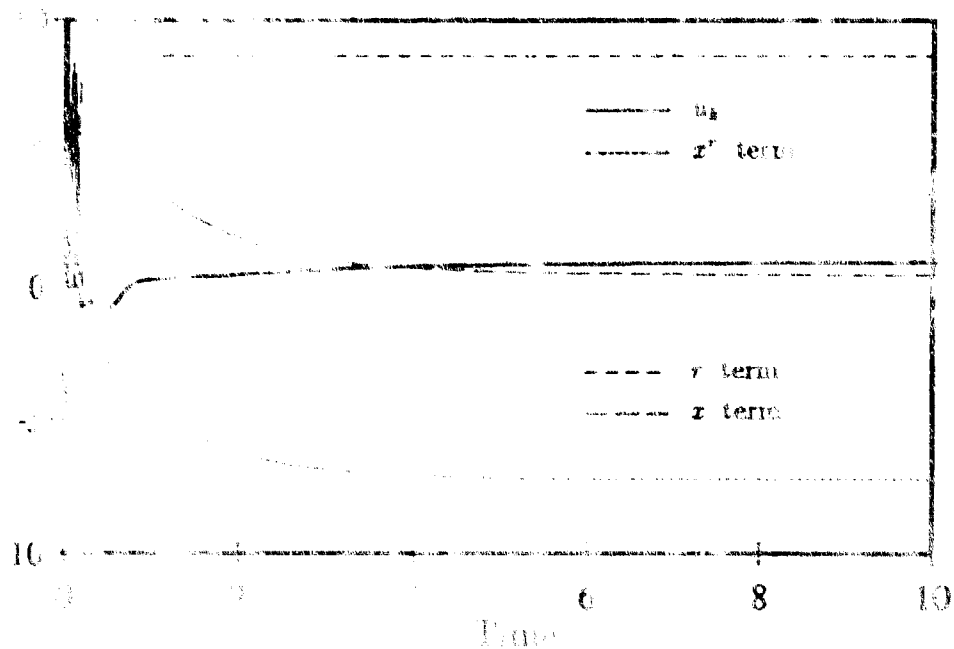


Figure 6.6. The control u_k and the constituent terms of the linear control law (6.1).

ATTACHMENT 3

6.3 Implementation and Results

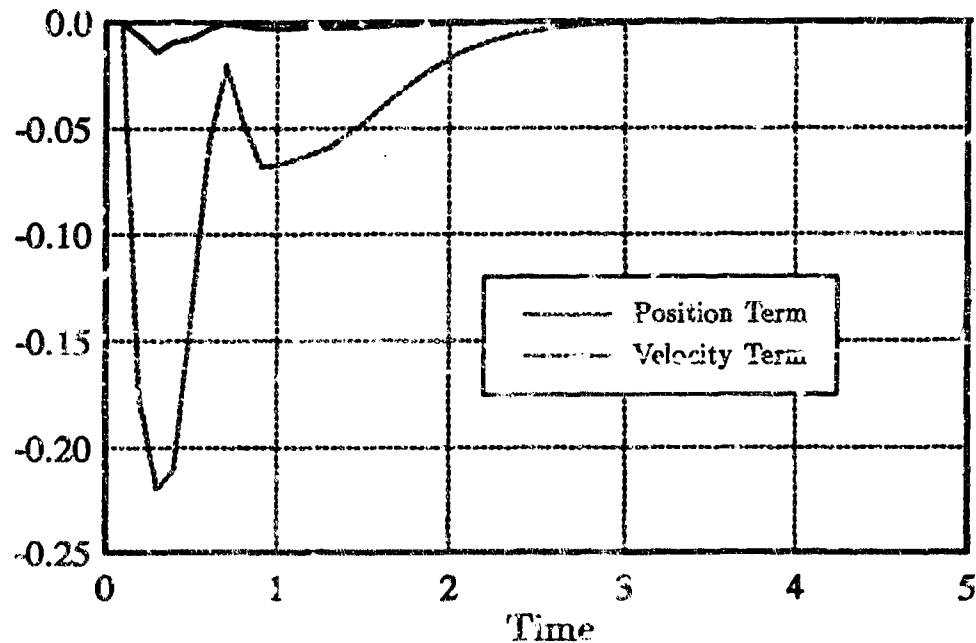


Figure 6.7. The estimated errors in the approximation of the initially unmodeled dynamics $f_k(x_k, u_{k-1})$.

tion of the nonlinear dynamics. These errors are most noticeable in the estimation of the velocity dynamics at velocities not near zero. Figure 6.8 illustrates the initial model errors not represented by the function f ; the adaptive term will reduce the effect of these remaining model errors. Experiments demonstrated that the system performed nearly as well when the adaptive contribution was removed from the control. A controller that augmented the linear law with only the adaptive correction was not evaluated.

Figure 6.8 shows the results of control laws (6.14) and (6.7) regulating the AEO from $x_0 = \{-1.0, 0.5\}$. The control magnitude was limited at 0.5 so that the results may be compared more easily to the benchmarks and the direct learning control results. Time is not explicitly shown in Figure 6.8; the state trajectory produced by

Chapter 6 - Indirect Learning Optimal Control

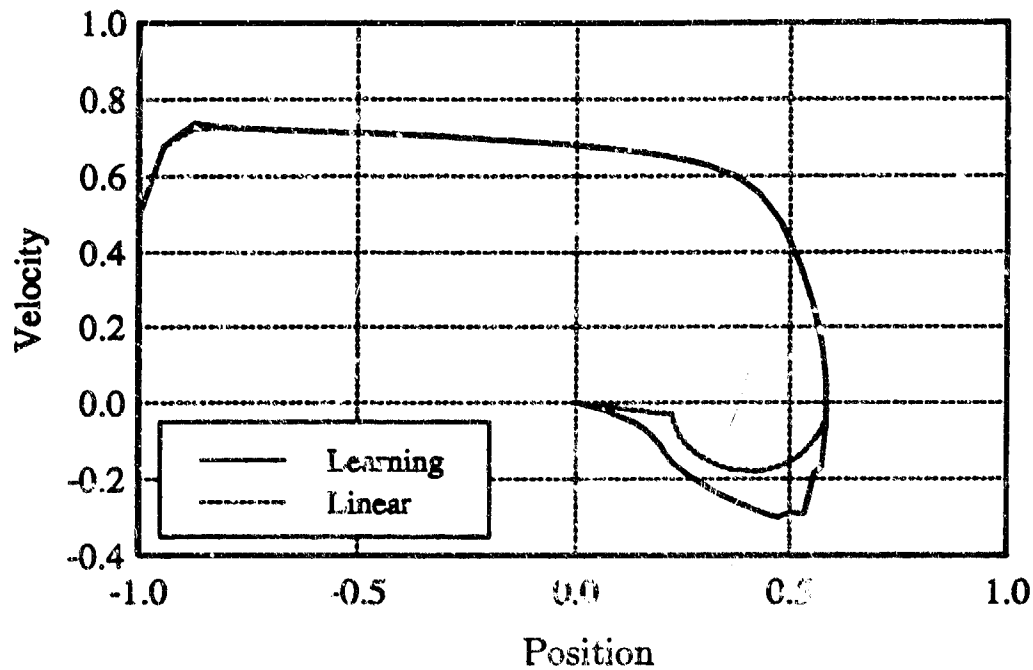


Figure 6.8. AEO Regulation from $x_0 = \{-1.0, 0.5\}$ with control saturation at ± 0.5 .

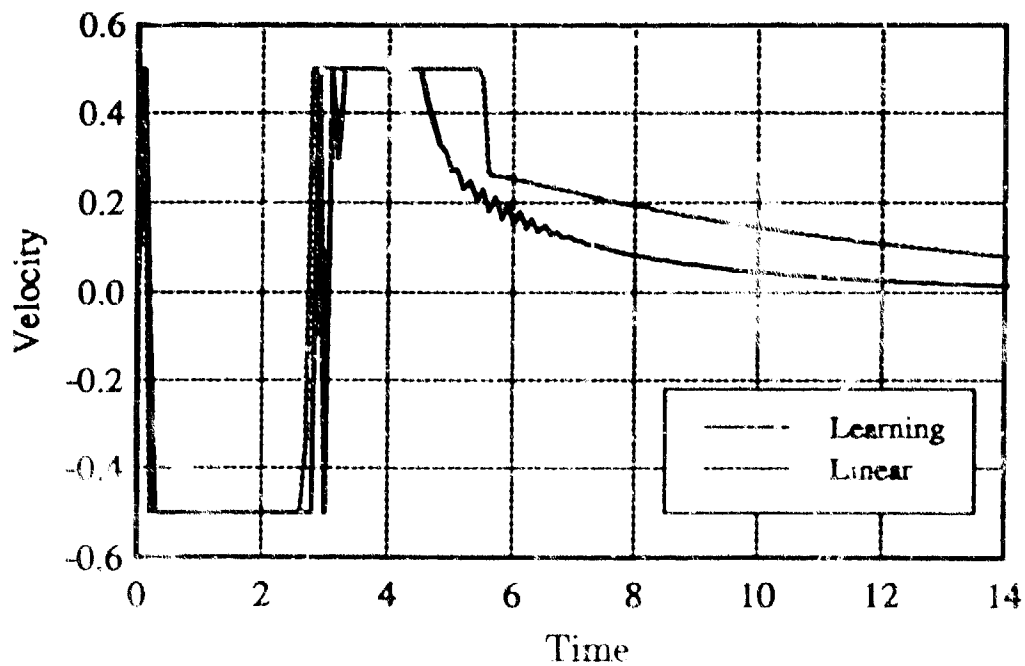


Figure 6.9. Control history associated with Figure 6.8.

6.3 Implementation and Results

the learning controller approached the origin much more quickly than the trajectory produced by the linear controller. The control objective remains to track a reference trajectory and, therefore, subtly differs from the goal of LQR (Figure 2.5). Recall that this reference model does not necessarily maximize system performance. Figure 6.9 shows the force histories which yielded the trajectories in Figure 6.8. The rapid switching in the learning control force results from learning errors and the sensitivity of the control law to the approximated Jacobian of $f_k(x_k, u_{k-1})$.

This indirect learning control technique was capable of learning, and therefore reducing the effect of, model uncertainty (linear and nonlinear). Therefore, the indirect learning controller derived from a linear model with model errors performed similar to Figure 6.8 and outperformed the LQR solution which was derived from an inaccurate linear model (Figure 2.7). The indirect learning controller with limited control authority produced state trajectories similar to the results of the direct learning control experiments.

Chapter 7

Summary

The aeroelastic oscillator demonstrated interesting nonlinear dynamics and served as an acceptable context in which to evaluate the capability of several direct and indirect learning controllers.

The ACP network was introduced to illustrate the biological origin of reinforcement learning techniques and to provide a foundation from which to develop the modified two-layer and single layer ACP architectures. The modified two-layer ACP introduced refinements that increased the architecture's applicability to the infinite horizon optimal control problem. However, results demonstrated that, for the defined plant and environment, this algorithm failed to synthesize an optimal control policy. Finally, the single layer ACP, which functionally resembled Q learning, successfully constructed an optimal control policy that regulated the aeroelastic oscillator.

Q learning approaches the direct learning paradigm from the mathematical theory of value iteration rather than from behavioral science. With sufficient train-

ATTACHMENT 3

Chapter 7 - Summary

ing, the Q learning algorithm converged to a set of Q values that accurately described the expected discounted future return for each state-action pair. The optimal policy that was defined by these Q values successfully regulated the aeroelastic oscillator plant. The results of the direct learning algorithms (e.g. the ACP derivatives and Q learning) demonstrated the limitations of optimal control laws that are restricted to discrete controls and a quantized input space. The concept of extending Q learning to accommodate continuous inputs and controls was considered. However, the necessary maximization at each time step of a continuous, potentially multi-modal Q function may render impractical an on-line implementation of a continuous Q learning algorithm.

The optimal control laws for single-stage and two-step finite time horizon, quadratic cost functionals were derived for linear and nonlinear system models. The results of applying these control laws to cause the AEO to optimally track a linear reference model demonstrated that indirect learning control systems, which incorporate information about the unmodeled dynamics that is incrementally learned, outperform fixed parameter, linear control laws. Additionally, operating with continuous inputs and outputs, indirect learning control methods provide better performance than the direct learning methods previously mentioned. A spatially localized connectionist network was employed to construct the approximation of the initially unmodeled dynamics that is required for indirect learning control.

7.1 Conclusions

This thesis has collected several direct learning optimal control algorithms and

7.1 Conclusions

has also introduced a class of indirect learning optimal control laws. In the process of investigating direct learning optimal controllers, the commonality between an algorithm originating in behavioral science and another founded in mathematical optimization help unify the concept of direct learning optimal control. More generally, this thesis has "drawn arrows" to illustrate how a variety of learning control concepts are related. Several learning systems were applied as controllers for the aeroelastic oscillator.

7.1.1 Direct / Indirect Framework

As a means of classifying approaches to learning optimal control laws, a *direct/indirect* framework was introduced. Both direct and indirect classes of learning controllers were shown to be capable of synthesizing optimal control laws, within the restrictions of the particular method being used. Direct learning control implies the feedback loop that motivates the learning process is closed around system performance. This approach is largely limited to discrete inputs and outputs. Indirect learning control denotes a class of incremental control law synthesis methods for which the learning loop is closed around the system model. The indirect learning control laws derived in §6 are not capable of yielding stable closed-loop systems for non-minimum phase plants.

As a consequence of closing the learning loop around system performance, direct learning control procedures acquire information about control saturation. Indirect learning control methods will learn the unmodeled dynamics as a function of the applied control, but will not "see" control saturation which occurs external to the control system.

Chapter 7 - Summary

7.1.2 Comparison of Reinforcement Learning Algorithms

The learning rules for the Adaptive Heuristic Critic (a modified TD(λ) procedure), Q learning, and Drive-Reinforcement learning (the procedure used in the ACP reinforcement centers) were compared. Each learning system was shown to predict an expected discounted future reinforcement. Moreover, each learning rule was shown to adjust the previous predictions in proportion to a prediction error that was the difference between the current reinforcement and the difference between the previous expected discounted future reinforcement and the discounted current expected discounted future reinforcement. The constants of proportionality describe the reduced importance of events that are separated by longer time intervals.

7.1.3 Limitations of Two-layer ACP Architectures

The limitations of the two-layer ACP architectures arise primarily from the simultaneous operation of two opposing reinforcement centers. The distinct positive and negative reinforcement centers, which are present in the two-layer ACP, incrementally improve estimates of the expected discounted future reward and cost, respectively. The optimal policy is to select, for each state, the action that maximizes the difference between the expected discounted future reward and cost. However, the two-layer ACP network performs reciprocal inhibition between the two reinforcement centers. Therefore, the information passed to the motor centers effects the selection of a control action that either maximizes the estimate of expected discounted future reward, or minimizes the estimate of expected discounted future cost. In general, a two-layer ACP architecture will not learn the optimal policy.

7.2 Recommendations for Future Research

7.1.4 Discussion of Differential Dynamic Programming

For several reasons, differential dynamic programming (DDP) is an inappropriate approach for solving the problem described in §1.1. First, the DDP algorithm yields a control policy only in the vicinity of the nominally optimal trajectory. Extension of the technique to construct a control law that is valid throughout the state space is tractable only for linear systems and quadratic cost functionals. Second, the DDP algorithm explicitly requires, as does dynamic programming, an accurate model of the plant dynamics. Therefore, for plants with initially unknown dynamics, a system identification procedure must be included. The coordination of the DDP algorithm with a learning systems that incrementally improves the system model would constitute an indirect learning optimal controller. Third, since the quadratic approximations are valid only in the vicinity of the nominal state and control trajectories, the DDP algorithm may not extend to stochastic control problems for which the process noise is significant. Fourth, similar to Newton's nonlinear programming method, the original DDP algorithm will converge to a globally optimal solution only if the initial state trajectory is sufficiently close to the optimal state trajectory.

7.2 Recommendations for Future Research

Several aspects of this research warrant additional thought. The extension of direct learning methods to continuous inputs and continuous outputs might be an ambitious endeavor. Millington [41] addressed this issue by using a spatially localized connectionist / Analog Learning Element that defined, as a distributed

ATTACHMENT 3

Chapter 7 - Summary

function of state, a continuous probability density function for control selection. The learning procedure increased the probability of selecting a control that yielded, with a high probability, a large positive reinforcement. The difficulty of generalizing the Q learning algorithm to continuous inputs and outputs has previously been discussed.

The focus of indirect learning control research should be towards methods of incremental function approximation. The accuracy of the learned Jacobian of the unmodeled dynamics critically impacts the performance of indirect learning optimal control laws. The selection of network parameters (e.g. learning rates, the number of nodes, and the influence function centers and spatial decay rates) determines how successfully the network will map the initially unmodeled dynamics. The procedure that was used for the selection of parameters was primarily heuristic. Automation of this procedure could improve the learning performance and facilitate the control law design process. Additionally, indirect learning optimal control methods should be applied to problems with a higher dimension. The closed-loop system performance as well as the difficulty of the control law design process should be compared with a gain-scheduled linear approach to control law design.

Appendix A

Differential Dynamic Programming

A.1 Classical Dynamic Programming

Differential dynamic programming (DDP) shares many features with the classical dynamic programming (DP). For this reason, and because dynamic programming is a more recognized algorithm, this chapter begins with a summary of the dynamic programming algorithm. R. E. Bellman introduced the classical dynamic programming technique, in 1957, as a method to determine the control function that minimizes a performance criterion [33]. Dynamic programming, therefore, serves as an alternative to the calculus of variations, and the associated two-point boundary value problems, for determining optimal controls.

Starting from the set of state and time pairs which satisfy the terminal conditions, the dynamic programming algorithm progresses backward in discrete time. To accomplish the necessary minimizations, dynamic programming requires a quantization of both the state and control spaces. At each discrete state, for every stage

ATTACHMENT 3

Appendix A - Differential Dynamic Programming

in time, the optimal action is the action which yields the minimum cost to complete the problem. Employing the principle of optimality, this completion cost from a given discrete state, for a particular choice of action, equals the sum of the cost associated with performing that action and the minimum cost to complete the problem from the resulting state [23]. $J_t^*(\underline{x})$ equals the minimum cost to complete a problem from state \underline{x} and discrete time t , $g(\underline{x}, \underline{u}, t)$ is the incremental cost function, where \underline{u} is the control vector, and $\underline{T}(\underline{x}, \underline{u}, t)$ is the state transition function. Further, define a mapping from the state to the optimal controls, $S(\underline{x}; t) = \underline{u}(t)$ where $\underline{u}(t)$ is the argument that minimizes the right side of (A.1).

$$J_t^*(\underline{x}(t)) = \min_{\underline{u}(t)} [g(\underline{x}(t), \underline{u}(t), t) + J_{t+1}^*(\underline{T}(\underline{x}(t), \underline{u}(t), t))] \quad (A.1)$$

The principle of optimality substantially increases the efficiency of the dynamic programming algorithm to construct $S(\underline{x}; t)$ with respect to an exhaustive search, and is described by Bellman and S. E. Dreyfus.

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [34].

The backward recursion process ends with the complete description of $S(\underline{x}; t)$ for all states and for $t = N-1, N-2, \dots, 1$, where N is the final time. Given the initial state $\underline{x}^*(1) = \underline{x}(1)$, (A.2) defines the forward DP recursion step.

$$\underline{u}^*(t) = S(\underline{x}^*; t) \quad (A.2a)$$

$$\underline{x}^*(t+1) = \underline{T}(\underline{x}^*, \underline{u}^*, t) \quad (A.2b)$$

ATTACHMENT 3

A.2 Differential Dynamic Programming

Although dynamic programming provides a general approach to optimal control problems, including the optimal control of nonlinear systems with state and control constraints, the DP algorithm requires substantial data storage and a large number of minimizations. The substantial data storage that dynamic programming requires results from the inefficient lookup table representation of J_t^* and \underline{u}^* at each quantized state and time; each item of data is represented exactly by a unique adjustable parameter. This *curse of dimensionality* also existed in the direct learning algorithms. Many practical problems, having fine levels of state and control quantization, require a continuous functional mapping, for which a single adjustable parameter encodes information over some region of the input space. Additionally, a continuous mapping eliminates the necessity to interpolate between discrete grid points in the input space to determine the appropriate control action for an arbitrary input. A learning system could be employed to perform this function approximation. A second disadvantage of the DP algorithm is the necessity of an accurate dynamic model. If the equations of motion are not accurately known a priori, explicit system identification is necessary to apply any dynamic programming procedure. The coordination of the DP algorithm with a learning system that incrementally improves the system model would constitute an indirect learning optimal controller.

A.2 Differential Dynamic Programming

Discrete time differential dynamic programming, introduced by D. Q. Mayne [35] and refined by D. H. Jacobson and Mayne [36], is a numeric approximation to

ATTACHMENT 3

Appendix A - Differential Dynamic Programming

the classical dynamic programming algorithm and is, therefore, also applicable to nonlinear discrete time optimal control problems.¹ Starting with a nominal state trajectory and a nominal control sequence, the DDP algorithm selects neighboring trajectories and sequences that yield the optimal decrease in the second-order approximation to the cost functional $J(\underline{u}) = \sum_{t=1}^N g(\underline{x}, \underline{u}, t)$.

The differential dynamic programming class of algorithms incorporates features of both dynamic programming and the calculus of variations. Before presenting an overview of the basic DDP algorithm, several of these properties will be reviewed. DDP does not involve the discretization of state and control spaces, which dynamic programming requires. Additionally, whereas dynamic programming constructs the value function of expected future cost to achieve the terminal conditions for each discrete state and each stage in discrete time, DDP constructs a continuous quadratic approximation to the value function for all states near the nominal trajectory. Finally, DDP solves for a control sequence iteratively, as do many solution techniques for the two-point boundary-value problems which arise from the calculus of variations. Bellman's algorithm (DP), in contrast, generates a control policy in a single computationally intensive procedure.

Each iteration of the differential dynamic programming algorithm consists of two phases: a backward run to determine $\delta \underline{u}(\underline{x}; t)$, the linear policy which defines the change from the nominal control as a function of state, for states near the nominal, and a forward run to update the nominal state trajectory and nominal control sequence [37,38]. The DDP algorithm requires accurate models of the incremental

¹ Jacobson and Mayne also applied the differential dynamic programming method to continuous time systems [36].

ATTACHMENT 3

A.2 Differential Dynamic Programming

cost function $g(\underline{x}, \underline{u}, t)$ and the state transition function $\underline{T}(\underline{x}, \underline{u}, t)$. Furthermore, the original DDP algorithm requires that both of these functions are twice differentiable with respect to states and controls; this condition is relaxed to a necessary single differentiability in several modified DDP algorithms.

The following development of the original DDP algorithm follows primarily from Yakowitz [39]. The nominal control sequence \underline{u}_n along with the initial state $\underline{x}(1)$ defines a nominal state trajectory \underline{x}_n .

$$\underline{u}_n = \{\underline{u}_n(1), \underline{u}_n(2), \dots, \underline{u}_n(N)\} \quad (A.3a)$$

$$\underline{x}_n = \{\underline{x}_n(1), \underline{x}_n(2), \dots, \underline{x}_n(N)\} \quad (A.3b)$$

The backward recursion commences at the final decision time, N , by constructing a quadratic approximation to the nominal cost.

$$L(\underline{x}, \underline{u}, N) = QP[g(\underline{x}, \underline{u}, N)] \quad (A.4)$$

The $QP[\cdot]$ operator selects the quadratic and linear, but not the constant, terms of the Taylor's series expansion of the argument about the nominal state and control sequences. A first order necessary condition for a control \underline{u}^* to minimize $L(\underline{x}, \underline{u}, N)$ appears in (A.5), which can be solved for the optimal input.

$$\Delta_{\underline{u}} L(\underline{x}, \underline{u}, N) = 0 \quad (A.5)$$

$$\delta \underline{u}(\underline{x}, N) = \underline{u}^*(N) - \underline{u}_n(N) = \alpha_N + \beta_N \delta \underline{x}(N) \quad (A.6a)$$

ATTACHMENT 3

Appendix A - Differential Dynamic Programming

$$\delta \underline{x}(N) = \underline{x}^*(N) - \underline{x}_n(N) \quad (A.6b)$$

The optimal value function, $f(\underline{x}, N) = \min_{\underline{u}} [g(\underline{x}, \underline{u}, N)]$, is also approximated by a quadratic.

$$V(\underline{x}; N) = L(\underline{x}, \underline{u}(\underline{x}, N), N) \quad (A.7)$$

The DDP backward calculations proceed for $t = N-1, N-2, \dots, 1$. Assuming that $V(\underline{x}; t+1)$ has been determined, the cost attributed to the current stage together with the optimal subsequent cost to achieve the terminal conditions is represented by $L(\underline{x}, \underline{u}, t)$.

$$L(\underline{x}, \underline{u}, t) = QP [g(\underline{x}, \underline{u}, t) + V(\underline{x}, \underline{u}, t; t+1)] \quad (A.8)$$

The necessary condition $\Delta_{\underline{u}} L(\underline{x}, \underline{u}, t) = 0$ yields the policy for the incremental control.

$$\delta \underline{u}(\underline{x}; t) = \underline{\alpha}_t + \beta_t(\underline{x}(t) - \underline{x}_n(t)) \quad (A.9)$$

$$\underline{u}(\underline{x}, t) = \underline{x}_n(t) + \delta \underline{u}(\underline{x}; t) \quad (A.10)$$

The expression for the variation in control (A.9) is valid for any state $\underline{x}(t)$ sufficiently close to the nominal state $\underline{x}_n(t)$. The vector $\underline{\alpha}_t$ and matrix β_t , $1 \leq t \leq N$, must be maintained for use in the forward stage of DDP. The optimal value function appears in (A.11).

$$V(\underline{x}; t) = L(\underline{x}, \underline{u}(\underline{x}, t), t) \quad (A.11)$$

The forward run calculates a successor control sequence and the corresponding state trajectory. Given $\underline{x}(1)$, $\underline{u}^*(1) = \underline{u}_n(1) + \underline{\alpha}_1$ by (A.9) and (A.10). Therefore,

ATTACHMENT 3

A.2 Differential Dynamic Programming

$\underline{x}^*(2) = \underline{T}(\underline{x}(1), \underline{u}^*(1), 1)$. For $t = 2, 3, \dots, N$, (A.12) defines the new control and state sequences which become the nominal values for the next iteration.

$$\underline{u}^*(\underline{x}, t) = \underline{\delta u}(\underline{x}(t), t) + \underline{u}_n(t) \quad (\text{A.12a})$$

$$\underline{x}^*(t+1) = \underline{T}(\underline{x}^*(t), \underline{u}^*(t), t) \quad (\text{A.12b})$$

The reduction of required computations, which differential dynamic programming demonstrates with respect to conventional mathematical programming algorithms, is most noticeable for problems with many state and control variables and many stages in discrete time. Whereas each iteration of the DDP algorithm involves solving a low dimensional problem for each stage in time, mathematical programming schemes for the numerical determination of an optimal trajectory typically require the solution of a single high dimensional problem for each iteration. To quantify this relationship, consider the problem where the state vector is a member of R^n , the control vector lies in R^m , and N represents the number of stages in discrete time. The DDP algorithm inverts N matrices of order m , for each iteration; the computational effort, therefore, grows linearly with N .² The method of variation of extremals provides a numeric solution to two-point boundary-value problems [23]. A single iteration of Newton's method for determining the roots of nonlinear equations, a technique for implementing the variation in extremals algorithm, in contrast, requires an $N \cdot m$ matrix to be inverted; the cost of an iteration, therefore, grows in proportion to N^3 [40]. Furthermore, both algorithms are quadratically convergent. In the case where $N = 1$, however, the DDP algorithm and Newton's method define identical incremental improvements in state and

² The control sequence will be in $R^{N \cdot m}$

ATTACHMENT 3

Appendix A - Differential Dynamic Programming

control sequences. [39] Similar computational differences exist between the DDP algorithm and other iterative numerical techniques such as the method of steepest descent and quasilinearization [23].

Appendix B

An Analysis of the AEO Open-loop Dynamics

This analysis follows directly from Parkinson and Smith [9]. Equation (2.12) may be written in the form of an ordinary differential equation with small nonlinear damping.

$$\frac{d^2 X}{d\tau^2} + X = \mu f\left(\frac{dX}{d\tau}\right) \quad \text{where} \quad \mu = nA_1 \ll 1 \quad (B.1)$$

If $\mu = 0$, the solution is a harmonic oscillator with a constant maximum vibration amplitude \bar{X} and phase ϕ .¹

$$X = \bar{X} \cos(\tau + \phi) \quad (B.2a)$$

$$\frac{dX}{d\tau} = -\bar{X} \sin(\tau + \phi) \quad (B.2b)$$

If μ is non-zero but much less than one ($0 < \mu \ll 1$) the solution may be expressed by a series expansion in powers of μ .

$$X = \bar{X} \cos(\tau + \phi) + \mu g_1(\bar{X}, \tau, \phi) + \mu^2 g_2(\bar{X}, \tau, \phi) + \dots \quad (B.3)$$

¹ All quantities in this analysis are nondimensional.

ATTACHMENT 3

Appendix B - Analysis of AEO Open-loop Dynamics

In the expansion, \bar{X} and ϕ are slowly varying functions of τ . To first-order, this series may be approximated by (B.2), where \bar{X} and ϕ are now functions of τ . For slowly varying \bar{X} and ϕ , these equations predict nearly circular trajectories in the phase plane. The parameters presented in §2.2 and used for all AEO experiments do not strictly satisfy these assumptions. However, the analysis provides insights to the AEO dynamics.

Following the outline presented in [9], each side of (B.1) is multiplied by \dot{X} and the algebra is manipulated.

$$(\ddot{X} + X)\dot{X} = \mu\dot{X}f(\dot{X}) \quad (B.4)$$

$$(\ddot{X} + X)\dot{X} = \frac{1}{2} \frac{d}{d\tau} (X^2 + \dot{X}^2) \quad (B.5)$$

$$X^2 + \dot{X}^2 = \bar{X}^2 \cos^2(\tau + \phi) + \bar{X}^2 \sin^2(\tau + \phi) = \bar{X}^2 \quad (B.6)$$

$$\mu\dot{X}f(\dot{X}) = -\mu\bar{X}\sin(\tau + \phi)f(-\bar{X}\sin(\tau + \phi)) \quad (B.7)$$

$$\frac{1}{2} \frac{d\bar{X}^2}{d\tau} = -\mu\bar{X}\sin(\tau + \phi)f(-\bar{X}\sin(\tau + \phi)) \quad (B.8)$$

That \bar{X} varies slowly with τ implies that the cycle period is small compared with the time intervals during which appreciable changes in the amplitude occur. Therefore, an average of the behavior over a single period eliminates the harmonics and is still sufficient for the purpose of examining the time evolution of the amplitude.

$$\frac{1}{2} \frac{d\bar{X}^2}{d\tau} = -\mu \frac{1}{2\pi} \int_0^{2\pi} \bar{X} \sin(\tau + \phi) f(-\bar{X}(\tau + \phi)) d\tau \quad (B.9)$$

ATTACHMENT 3

Recall from (2.12) and (B.1) that $f(\frac{dX}{d\tau})$ is given by

$$f\left(\frac{dX}{d\tau}\right) = \frac{1}{1000} \left[1000 \left(U - \frac{2\beta}{nA_1} \right) \frac{dX}{d\tau} - \left(\frac{A_3}{A_1 U} \right) \left(1000 \frac{dX}{d\tau} \right)^3 + \left(\frac{A_5}{A_1 U^3} \right) \left(1000 \frac{dX}{d\tau} \right)^5 - \left(\frac{A_7}{A_1 U^5} \right) \left(1000 \frac{dX}{d\tau} \right)^7 \right]. \quad (B.10)$$

Therefore, (B.9) reduces to

$$\frac{d\bar{X}^2}{d\tau} = \frac{nA_1}{1000} \left[1000 \left(U - \frac{2\beta}{nA_1} \right) \bar{X}^2 - \frac{3}{4} \left(\frac{A_3}{A_1 U} \right) 1000^3 \bar{X}^4 + \frac{5}{8} \left(\frac{A_5}{A_1 U^3} \right) 1000^5 \bar{X}^6 - \frac{35}{64} \left(\frac{A_7}{A_1 U^5} \right) 1000^7 \bar{X}^8 \right]. \quad (B.11)$$

In the following analysis, let R represent the square of the amplitude of the reduced vibration, i.e. $R = \bar{X}^2$. Equation (B.11) may immediately be rewritten in terms of R .

$$\frac{dR}{d\tau} = aR - bR^2 + cR^3 - dR^4 \quad (B.12)$$

Recalling that $\mu \ll 1$, stationary oscillations are nearly circular and correspond to constant values of \bar{X}^2 ; constant values of \bar{X}^2 are achieved when

$$\frac{dR}{d\tau} = 0. \quad (B.13)$$

This condition is satisfied by $R = 0$ and also by the real, positive roots of the cubic $a - bR + cR^2 - dR^3 = 0$. Negative or complex values for the squared amplitude of vibration would not represent physical phenomena.

ATTACHMENT 3

Appendix B - Analysis of AEO Open-loop Dynamics

Stability is determined by the tendency of the oscillator to converge or diverge in response to a small displacement δR . The sign of $\left. \frac{d}{dR} \left(\frac{dR}{d\tau} \right) \right|_{R=R_i}$ determines the stability of the focus and the limit cycles and will be positive, negative, or zero for unstable, stable, and neutrally stable trajectories, respectively.

$$\frac{d}{dR} \left(\frac{dR}{d\tau} \right) = a - 2bR + 3cR^2 - 4dR^3 \quad (B.14)$$

The stability of the focus is easily analyzed.

$$\left. \frac{d}{dR} \left(\frac{dR}{d\tau} \right) \right|_{R=0} = a = nA_1 \left(U - \frac{2\beta}{nA_1} \right) \quad (B.15)$$

$$A_1 = \left. \frac{dC_l}{d\alpha} \right|_{\alpha=0} > 0 \quad (B.16)$$

Given that n , U , A_1 , A_3 , A_5 , and A_7 are positive, the coefficients b , c , and d will also be positive. If $\beta = 0$, the system has no mechanical damping and a will be positive for all values of windspeed. However, if $\beta > 0$, then $a > 0$ only if $U > U_c = \frac{2\beta}{nA_1}$. Therefore, if $\beta = 0$ the focus is unstable for all windspeeds greater than zero, and if $\beta > 0$ the focus is unstable for $U > U_c$. This minimum airspeed for oscillation is the definition of the reduced critical windspeed; oscillation can be eliminated for windspeeds below a specific value by sufficiently increasing the mechanical damping.

Three distinct solutions exist when $a > 0$; the focus is unstable for each. The choice among these possibilities, which are characterized by the real positive roots of the cubic $a - bR + cR^2 - dR^3 = 0$, depends upon the windspeed. (1) If R_1 is the

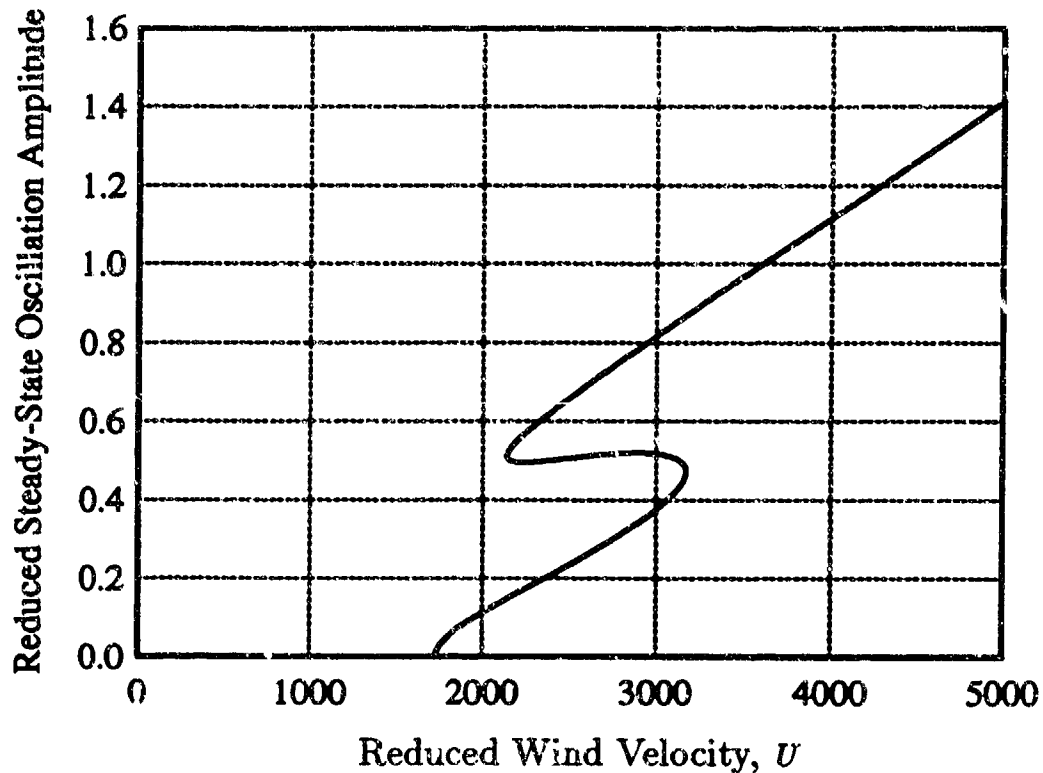


Figure B.1. The steady state amplitudes of oscillation \bar{X}_{ss} versus the incident windspeed U .

single real, positive root, there is a single stable limit cycle of radius $\sqrt{\bar{R}_1}$ around the unstable focus. This condition exists for two ranges of the reduced windspeed. (2) Three real, distinct, positive roots, $R_3 > R_2 > R_1$, correspond to two stable limit cycles at $\sqrt{\bar{R}_1}$ and $\sqrt{\bar{R}_3}$ separated by an unstable limit cycle at $\sqrt{\bar{R}_2}$. (3) R_1 and R_2 coalesce to a double, real, positive root while R_3 is a distinct, real, positive root. The magnitude of the radius of the single stable limit cycle depends on prior state information; this hysteresis is discussed below. This condition occurs at two values of the reduced incident windspeed.

The most interesting dynamics occur when the second of these situations exists. Figure B.1 plots the steady state amplitude of oscillation \bar{X}_{ss} , for circular

ATTACHMENT 3

Appendix B - Analysis of AEO Open-loop Dynamics

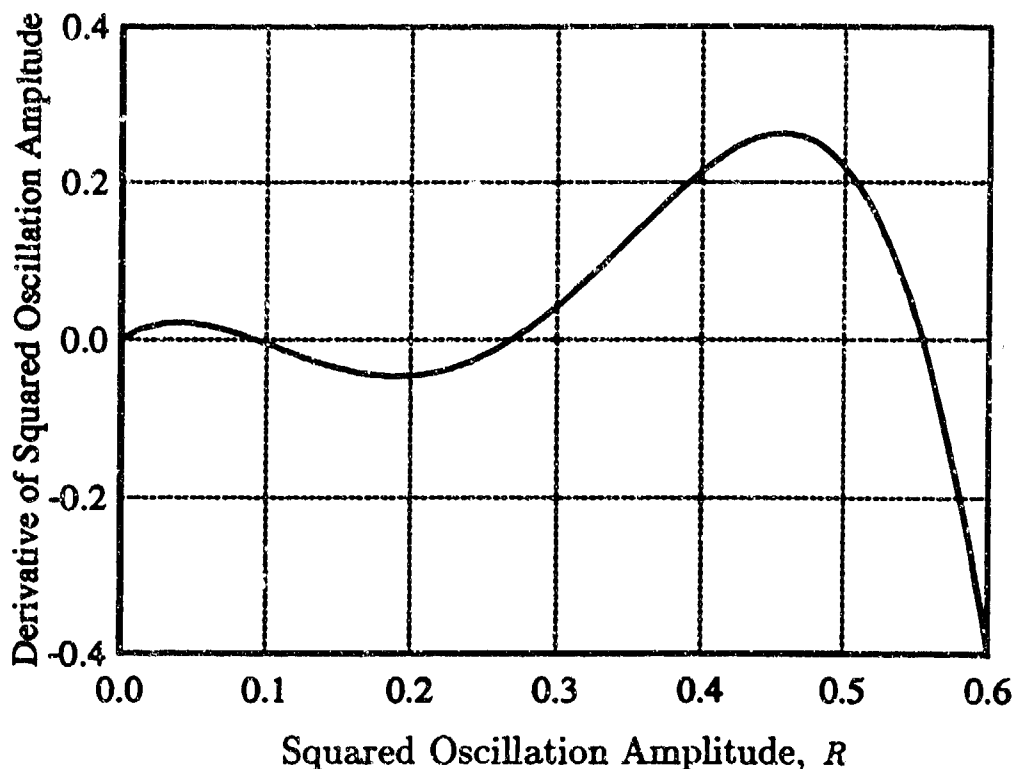


Figure B.2. $\frac{dR}{d\tau}$ versus R for $U = 2766.5$.

limit cycles, as a function of incident windspeed.

A hysteresis in \bar{X}_{ss} can be demonstrated by increasing the reduced airspeed from $U < U_c$, where $\bar{X}_{ss} = 0$. For $U_c < U < U_2$, the amplitude of the steady state oscillation will correspond to the inner stable limit cycle; for $U > U_2$, \bar{X}_{ss} jumps to the larger stable limit cycle. As the dimensionless windspeed is decreased from $U > U_2$, the amplitude of the steady state oscillation will remain on the outer stable limit cycle while $U > U_1$.² When the windspeed is decreased below $U = U_1$, the steady state amplitude of oscillation decreases to the inner stable limit cycle. Therefore, for a constant windspeed $U_1 < U < U_2$, \bar{X}_{ss} resides on the inner

² $U_c < U_1 < U_2$.

ATTACHMENT 3

stable limit cycle when the initial displacement is less than the magnitude of the unstable limit cycle, and \bar{X}_{ss} lies on the outer stable limit cycle when the initial displacement is greater than the magnitude of the unstable limit cycle.

For a specific value of the reduced wind velocity, the rate of change of the square of the oscillation amplitude, $\frac{dR}{dt}$, can be plotted against the square of the amplitude of oscillation R (Figure B.2). If $\frac{dR}{dt}$ is positive, the oscillation amplitude will increase with time, and if $\frac{dR}{dt}$ is negative the oscillation amplitude will decrease with time. Therefore, an oscillation amplitude where the value of $\frac{dR}{dt}$ crosses from positive to negative with increasing R is a stable amplitude. The focus will be stable when the time rate of change of oscillation amplitude is negative for R slightly greater than zero.

References

- [1] Sutton, Richard S., Andrew G. Barto, and Ronald J. Williams, "Reinforcement Learning is Direct Adaptive Optimal Control," *IEEE Control Systems Magazine*, vol. 12, no. 2, pp. 2143-2146, (1992).
- [2] Farrell, Jay and Walter Baker, "Learning Augmented Control for Advanced Autonomous Underwater Vehicles," Proceedings of *The 18'th Annual Symposium & Exhibit of the Association for Unmanned Vehicle Systems*, Washington D.C., (1991).
- [3] Baker, Walter L. and Jay A. Farrell, "Learning Augmented Flight Control for High Performance Aircraft," Proceedings of *AIAA GN&C Conference*, New Orleans, LA, (1991).
- [4] Baker, Walter and Jay Farrell, "An Introduction to Connectionist Learning Control Systems," *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive*, D. White and D. Sofge (eds.), Van Nostrand Reinhold, New York, pp. 35-64, (1992).
- [5] Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, (1983).
- [6] Barto, Andrew G. and P. Anandan, "Pattern-Recognizing Stochastic Learning Automata," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 3, pp. 360-375, (1985).
- [7] Millington, Peter J. and Walter L. Baker, "Associative Reinforcement Learning for Optimal Control," Proceedings of *The AIAA Guidance, Navigation, and Control Conference*, Portland, OR, (1990).
- [8] Michie, D. and R. A. Chambers, "BOXES: An Experiment in Adaptive Control," *Machine Intelligence 2*, Ella Dale and Donald Michie (eds.), American Elsevier Publishing Company, New York, pp. 137-152, (1968).

ATTACHMENT 3

References

- [9] Parkinson, G. V. and J. D. Smith, "The Square Prism as an Aeroelastic Non-linear Oscillator," *Quarterly Journal of Mechanics and Applied Mathematics*, vol. 17, pp. 225-239, (1964).
- [10] Thompson, J. M. T. and H. B. Stewart, *Nonlinear Dynamics and Chaos*, pp. 60-64, John Wiley and Sons, New York, (1986).
- [11] Alexander, Jeff, L. Baird, W. Baker, and J. Farrell, "A Design & Simulation Tool for Connectionist Learning Control Systems: Application to Autonomous Underwater Vehicles," *Proceedings of The Conference of the Society for Computer Simulation*, Baltimore, MD, (1991).
- [12] Klopff, A. Harry, "A Neuronal Model of Classical Conditioning," *Psychobiology*, vol. 16, no. 2, pp. 85-125, (1988).
- [13] Baird III, Leemon C. and A. Harry Klopff, "Extensions of the Associative Control Process (ACP) Network: Hierarchies and Provable Optimality," *Proceedings of the conference Simulation of Adaptive Behavior*, (1992).
- [14] Klopff, A. Harry, James S. Morgan, and Scott E. Weaver, "A Hierarchical Network of Control Systems that Learn: Modeling Nervous System Function During Classical and Instrumental Conditioning," Submitted to *Adaptive Behavior*, (1992).
- [15] Sutton, Richard S., "Learning to Predict by Methods of Temporal Differences," *Machine Intelligence 3*, Kluwer Academic Publishers, Boston, pp. 9-44, (1988).
- [16] Watkins, C., "Learning from Delayed Rewards," Doctoral thesis, Cambridge University, Cambridge, England, (1989).
- [17] Watkins, Christopher J. C. H. and Peter Dayan, "Q-Learning," *Machine Learning*, (1992).
- [18] Samuel, A. L., "Some Studies in Machine Learning Using the Game of Checkers," *Computers and Thought*, E. A. Feigenbaum and J. Feldman (eds.), McGraw Hill, New York, (1959).
- [19] Holland, J. H., "Escaping Brittleness: The Possibility of General-purpose Learning Algorithms Applied to Parallel Rule-based Systems," *Machine Learning: An Artificial Intelligence Approach*, vol. 2, R. S. Michalski, J. G. Carbonell, and J. M. Mitchell (eds.) Morgan Kaufmann, Los Altos, CA, (1986).

ATTACHMENT 3

- [20] Sutton, R. S., "Temporal Credit Assignment in Reinforcement Learning," Doctoral thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, (1984).
- [21] Widrow, B. and M. E. Hoff, "Adaptive Switching Circuits," *1960 WESCON Convention Record*, Part 4, pp. 96-104, (1960).
- [22] Bryson, Jr., Arthur E. and Yu-Chi Ho, *Applied Optimal Control*, Hemisphere Publishing Corp., New York, (1975).
- [23] Kirk, Donald E., *Optimal Control Theory*, Prentice-Hall Inc., Englewood Cliffs, NJ, (1970).
- [24] Anderson, Mark R. and David K. Schmidt, "Error Dynamics and Perfect Model Following with Application to Flight Control," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 14, no. 5, pp. 912-919, (1991).
- [25] Baker, Walter L. and Peter J. Millington, "Design and Evaluation of a Learning Augmented Longitudinal Flight Control System," *Proceedings of The 32'nd IEEE Conference on Decision and Control*, San Antonio, TX, (1993).
- [26] Press, W., B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, (1988).
- [27] Youcef-Toumi, K. and Ito Osamu, "A Time Delay Controller for Systems with Unknown Dynamics," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 112, (1990).
- [28] Nistler, Noel F., "A Learning Enhanced Flight Control System for High Performance Aircraft," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, (1992).
- [29] Franklin, G. F., J. D. Powell, and M. L. Workman, *Digital Control of Dynamic Systems*, Addison-Wesley, Reading, MA, (1980).
- [30] Funahashi, K. "On the Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, vol. 2, pp. 183-192, (1988).
- [31] Minsky, L. and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, (1969).
- [32] Poggio, T. and F. Girosi, "Networks for Approximation and Learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481-1497, (1990).
- [33] Bellman, R. E., *Dynamic Programming*, Princeton University Press, Princeton, NJ, (1957).

ATTACHMENT 3

References

- [34] Bellman, R. E., and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, (1962).
- [35] Mayne, D., "A Second-Order Gradient Method for Determining Optimal Trajectories of Nonlinear Discrete-Time Systems," *International Journal on Control*, vol. 3, pp. 85-95, (1966).
- [36] Jacobson, David H. and David Q. Mayne, *Differential Dynamic Programming*, American Elsevier Publishing Company, Inc., New York, (1970).
- [37] Lopez-Coronado, J. and L. Le Letty, "Differential Dynamic Programming — Implementation of Algorithms and Applications," *Simulation in Engineering Sciences*, J. Burger and Y. Jarny (eds.), Elsevier Science Publishers, B. V. North-Holland, pp. 93-102, (1983).
- [38] Sen, S. and S. J. Yakowitz, "A Quasi-Newton Differential Dynamic Programming Algorithm for Discrete-time Optimal Control," *Automatica*, vol. 23, no. 6, pp. 749-752, (1987).
- [39] Yakowitz, Sydney, "Algorithms and Computational Techniques in Differential Dynamic Programming," *Control and Dynamic Systems*, vol. 31, (1989).
- [40] Pantoja, J. F. A. DE O., "Differential Dynamic Programming and Newton's Method," *International Journal of Control*, vol. 47, no. 5, pp. 1539-1553, (1988).
- [41] Millington, Peter J., "Associative Reinforcement Learning for Optimal Control," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, (1991).
- [42] Baird, Leemon C., "Function Minimization for Dynamic Programming Using Connectionist Networks," Proceedings of *The IEEE Conference on Systems, Man, and Cybernetics*, Chicago, IL, (1992).